[Continued on next page]

(54) Title: DIRECT RESPONSE E-MAIL

(57) Abstract: An e-mail message [10] is analyzed to derive response information concerning a commercial transaction. Based on the derived information, commercial transaction data are automatically generated in a format that is usable to complete the transaction.

WILLIAM HERP   /10

FROM: HARVARD BUSINESS SCHOOL PUBLISHING
SENT: WEDNESDAY, DECEMBER 16, 1998 9:19 PM
TO: WILLIAM HERP
SUBJECT: ━ NEW INSIGHTS FROM HARVARD BUSINESS REVIEW

HARVARD BUSINESS SCHOOL PUBLISHING CORPORATION
BOSTON, MASSACHUSETTS USA

THURSDAY, DECEMBER 17, 1998

DEAR WILLIAM HERP,

ON THURSDAY, DECEMBER 3RD, WE WROTE YOU REGARDING A SPECIAL OFFER ON THE HARVARD BUSINESS REVIEW PAPERBACK SERIES. SINCE WE HAVE NOT HEARD BACK, WE WANTED TO FOLLOW UP BEFORE THIS SPECIAL OFFER CLOSES. IF YOU ARE SIMPLY NOT INTERESTED, WE APOLOGIZE FOR THE INTRUSION. BELOW PLEASE FIND THE ORIGINAL OFFER IN ITS ENTIRETY.

THE HARVARD BUSINESS REVIEW PAPERBACK SERIES BRINGS YOU THE LATEST AND MOST SIGNIFICANT THINKING ON TODAY'S MOST PRESSING MANAGEMENT CHALLENGES. THESE INSIGHTFUL COLLECTIONS ARE THE DEFINITIVE RESOURCE FOR PROFESSIONALS.

EACH TITLE:
   + PROVIDES A BROAD UNDERSTANDING OF AN ISSUE
   + IS CLEARLY WRITTEN AND, IN MANY CASES, DRAWS UPON REAL COMPANY EXAMPLES
   + HELPS YOU CONSTRUCT A USEFUL CONCEPTUAL FRAMEWORK FOR DECISION-MAKING AND IMPLEMENTATION
   + CONTAINS EIGHT ARTICLES FROM HARVARD BUSINESS REVIEW   ⎫
                                                                                   ⎬ 12
◇ EACH PAPERBACK IS $19.95 PLUS SHIPPING AND HANDLING ◇   ⎭
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

TO ORDER ONE OR MORE OF THESE PAPERBACKS, SIMPLY REPLY TO THIS MESSAGE AND NOTE THE LETTER (A-F) OF THE HARVARD BUSINESS REVIEW PAPERBACK YOU WOULD LIKE TO RECEIVE. PLEASE TYPE THE LETTERS IN THE 1ST LINE OF THE BODY OF YOUR REPLY E-MAIL. SHIPPING AND HANDLING CHARGES WILL BE APPLIED TO EACH ORDER.

YOUR CHOICES (DETAILED BELOW) ARE:

A: HARVARD BUSINESS REVIEW ON CHANGE PAPERBACK
B: HARVARD BUSINESS REVIEW ON KNOWLEDGE MANAGEMENT PAPERBACK
C: HARVARD BUSINESS REVIEW ON STRATEGIES FOR GROWTH PAPERBACK   ⎫ 14
D: HARVARD BUSINESS REVIEW ON MEASURING CORPORATE PERFORMANCE PAPERBACK ⎬
E: HARVARD BUSINESS REVIEW ON LEADERSHIP PAPERBACK   ⎭

F: THE EXECUTIVE COLLECTION - ALL FIVE TITLES FOR $89

OR IF YOU PREFER, CALL 1-800-668-6780 (617-496-1449 OUTSIDE THE U.S.)
MON. - FRI. 8 A.M. - 6 P.M. EST. PLEASE BE SURE TO MENTION PRIORITY CODE 1202.
>>

WO 01/06435 A1

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

DIRECT RESPONSE E-MAIL

Field of invention

5   This invention relates to direct response e-mail.

Background of the Invention

In direct response e-mail, a vendor, for example, can sell a product to a customer by sending an e-mail message to the customer that describes the product and its

10   price. The customer can order the product by returning an e-mail (sometimes called a direct response e-mail) that gives appropriate order information. The vendor can confirm the order by a return e-mail. The order information returned by the customer can sometimes be determined

15   automatically using software that analyses the customer's reply e-mail.

Summary of the Invention

In general, in one aspect of the invention, an e-mail message is analyzed to derive response information

20   concerning a commercial transaction. Based on the derived information, commercial transaction data are automatically generated in a format that is usable to automatically complete the commercial transaction.

In general, in another aspect of the invention, an

25   e-mail message is sent to a customer offering a product or service for sale. The e-mail message includes locations for response by the customer to indicate his intention to order the product or service. The customer returns an e-mail message that includes the response. Based on the received

30   e-mail, order information is automatically generated in a format usable automatically by an order fulfillment system to cause the order to be filled.

In general, another aspect of the invention includes automatically identifying response information which

requires resolution of an issue with the source of the e-mail message and automatically managing an e-mail dialog with the source to resolve the issue.

In general, in another aspect, the invention features automatically sorting e-mail messages, based on response information contained in the messages, into e-mail messages that can be processed automatically to generate commercial transactions, e-mail messages in which the response information is inadequate to permit generation of commercial transactions, and e-mail messages that may be subjected to exception handling to yield information that is sufficient to generate commercial transactions.

In general, in another aspect, the invention features automatically generating a confirmatory e-mail message to the source of the e-mail message confirming that a commercial transaction has been or will be completed.

In general, in another aspect, the invention features receiving inbound e-mail messages that result from corresponding outbound e-mail messages associated with a marketing program, the inbound messages containing response information, each of the outbound messages being associated with a distinct piece of the marketing program. The response information in each of the inbound messages is automatically associated with the corresponding distinct piece of the marketing program.

In general, in another aspect, the invention features automatically merging response information with corresponding information in a database for use in completing transactions.

In general, in another aspect, the invention features identifying inbound e-mail messages that cannot be processed automatically to generate commercial transactions,

2

and using the database information to assist in exception
handling of the identified inbound messages.

Other advantages and features will become apparent
from the following description and from the claims.

5   Brief Description of the Drawing
Figures 1A through 1C and 2A through 2B show e-mail
messages.

Figure 3 is a block diagram of a direct response e-
mail system.

10   Description of the Preferred Embodiments
Outbound e-mail messages
The two e-mail messages shown in figures 1A through
1C and 2A through 2B are examples of outbound messages
associated with commercial transactions.

15   The example message 10 shown in figure 1 offers
Harvard Business Review products. Message 10 includes basic
copy 12 that is similar to basic direct marketing copy of
the kind that is commonly used in e-mail marketing. Message
10 also contains a section 14 giving instructions on how to

20   order the products.
Inbound e-mail messages
To take advantage of the offer shown in figure 1,
the recipient creates a reply e-mail message (the direct
response message) and types the letters of the items that he

25   wants to order in the first line of the body of the message.
In other examples, the letters could be typed in the subject
line or the last line of the body of the message. The user
is also asked to correct and complete shipping and e-mail
address information that has been merged into the outbound

30   e-mail message in a section 16. In section 16, each of the
entries is bounded by brackets. Another section could

3

contain merged billing information, not shown. The person who replies to the e-mail (the customer) is meant to include the corrections or additions within the indicated brackets.

5   By allowing the recipient to take advantage of the offer simply by replying to the e-mail, rather than requiring the recipient to place an order by linking to a related web-site or to print the e-mail message and FAX it back, or to call an 800 number, a much higher return rate can be achieved. For conventional outbound e-mail messages

10   that require the recipient to click on an embedded URL to go to a web site, the returns may be on the order of several hundred percent on investment (the fee charged for delivering the outbound messages). By enabling the recipient to provide direct response e-mail messages as in

15   figure 1, the return on investment can be as high as several thousand percent.

       Figures 2A and 2B illustrate a similar outbound e-mail message in which there is no choice of products but only a single offer to be accepted or rejected. To take

20   advantage of the offer, the recipient types "yes" in the subject line. In figure 2B, a shipping block 18 of the kind mentioned above is shown. (In this case, the shipping block contains no information because the shipping address is the same as the billing address.)

25   One reason for including differential billing and shipping blocks is to acquire information in the return e-mail message that is similar to information captured in orders placed on a related web site. In a system in which web-site orders generate fields that can be fed directly to

30   an automated order fulfillment process, it is useful to make the e-mail message information field-wise consistent to permit the information to be delivered automatically to the same order fulfillment process.

Exception processing

Processing the inbound e-mails (the ones with responses concerning commercial transactions from the recipients of the outbound e-mails) may require custom

5   interaction with the recipients. For example, the wording of the outbound messages may be confusing to the recipients.

As shown in figure 3, the system 40 enables the transactional e-mail message processor 42 to determine when a dialog with the recipient 44 is needed and then assists a

10   human service representative 46 to conduct an effective dialog 48. The dialog can be conducted on behalf of the vendor 50 but without involving the vendor. Alternatively, the vendor's fulfillment process 52 can be notified electronically 54 of interaction that may be required.

15   Easing the processing of responses that include customer orders is important because the orders typically come back quickly, e.g., within 36-48 hours, and in large volume. The ability to deal with questions that arise as a result of the contact from a customer service point of view keeps the

20   vendor's customer service organization from being overwhelmed by the responses that come back.

The ability to process exceptions without involving the customer service organization of the vendor is based partly on knowing how the outbound e-mail messages were

25   constructed. As a simple example, a recipient may ask an unnecessary question that could have been answered by reading the outbound e-mail message. The e-mail message processor can pull out the relevant portion of the message and send it back to the recipient to answer the question.

ProcOrder Process

The inbound e-mail messages 60 are batch processed
by a script called ProcOrder 62. ProcOrder parses the
elements of the inbound e-mail messages in accordance with
5 the original set up and instructions of the outbound e-mail
messages 64. ProcOrder determines if all of the items that
are required for an order to be completely processed
automatically appear in the inbound e-mail message. For
example, the script would look for the ordering token, such
10 as the word "yes" or a series of letters depending on
whether it is a single or multiple offer. The script would
also look for footer information in the e-mail message,
including a code that identifies the given campaign and the
given offer, as seen in block 66 of figure 2B. In that
15 example, there are four components in the footer, but only
two are represented because the other two are not required
in this instance. The first element is a customer
identifier 68, e.g. 861270. Then there is a space 70
between two pipes that would contain the list identifier if
20 there were one. There may be multiple recipient lists for a
given marketing campaign. In the example, there is only one
list, and there is no list identifier. A list number 243
might refer to a list of people who made a purchase at the
vendor's web site or who subscribed at the web-site for a
25 listserv.

The third footer item could be a source of awareness
code 72, e.g., 3275, which identifies a particular marketing
campaign. For example, in the case of figure 2, the code
could refer to a Benchmarking Three-part Video Series offer.
30 The last item in the footer, located between the
final pipe and the first right bracket would be a flight
identification code 74. A given campaign could have
multiple flights of e-mail messages.

After looking for the footer information, the
ProcOrder parser looks for fields in the billing and
shipping address blocks that are required to complete the
order. What is required may vary with the type of campaign
5   but typically the minimum requirements are a name and a
physical address. If the information is not completely
available in the response e-mail message, the script checks
to see if it is available in the database 76. If not
available in either place, the script generates an exception
10 entry for an exception list. The exception list is provided
to a service representative 46 who can then act on it
(without involving the vendor's customer service
organization), e.g., by sending back an e-mail message
asking for the shipping address.

15   If all required information is available, the script
generates a fully fielded valid order in a format required
by the fulfillment system of the vendor and adds it to a
batch of valid orders 78 which are sent electronically to
the fulfillment process.

20 Confirmation e-mail message

As a result of running the ProcOrder script, an e-
mail message 80 is returned to each customer either to
confirm an order or to request more information. In the
latter case, a dialog ensues and is managed by software and
25 through an exception handling service as explained earlier.
For example, the customer's response could say something
like "sure, send"; or "send it and I'll take a look."
Shortly thereafter the customer would receive a confirmation
"Thank you for your order; you can expect the CD-ROM in
30 about seven business days. Please let us know if there is
anything else we can do to help simply by replying to this
e-mail."

One-click ordering

Another feature of the e-mail dialog with a customer
involves simplifying and optimizing the presentation of
content. In the examples of figures 1 and 2, the
5   information is presented in a simple text format. It is
useful also to provide in-line HTML code in the outbound e-
mail message in a manner similar to the one-click ordering
that Amazon.com offers in a web-site context. In one-click
ordering, the customer sets up an account by providing
10 credit card and shipping information. On subsequent visits
to the web site, the customer can pick a product with one
click, place an order, and have it shipped. A similar
technique could be adapted to e-mail message interchange by
embedding one-click ordering into e-mail.

15   An advantage of in-line HTML code is the opportunity
for a much higher response rate because of the higher
graphical contact and higher level of engagement normally
achieved by a graphical message.

Template

20   The outbound e-mail messages are set up in a
standard format using templates 90. The templates enable
either a single-offer message or a multiple-offer message.
Other templates are also possible, including one that embeds
in-line HTML into the message as mentioned above, either for
25 the single-offer or multiple-offer cases.

In addition, a set-up tool 92 permits the parameters
of a given campaign to be defined, including the source of
awareness code, the flight identification code, the campaign
identification code, and similar information. The set-up
30 tool also permits defining the tokens that are to be used in
a given campaign (for example, the letters assigned to
different products being offered). The set-up tool also
allows a definition of the required fields that must appear

in a given campaign to enable automated generation of orders
to an existing fulfillment system.

The set-up tool also provides a user interface that
enables a vendor to help in entering the set-up information.

5   The result of applying the tool to the templates is
a set of outbound message forms 94 that are ready for use.

Reporting tool

After the template is set up and the system is ready
to launch a flight, address 108 and other information 110.

10   112 stored in the target list of customers is merged with
the message forms, and the e-mail messages are automatically
generated and sent by an outbound e-mail delivery engine 96.
Customers then begin to respond. The ProcOrder script
generates automatic orders to the fulfillment system and

15   exception information for additional processing.

A reporting tool 104 aggregates information about
the responses for a given campaign according to source of
awareness code and flight. The information is made
available on-line to the vendor and can be used for a

20   variety of marketing purposes. The information could be
generated as an Excel file attached to an e-mail, or as a
paper-based report, or as an electronic file that is
transferred on a batch basis.

Gathering additional information from database

25   There may be an intermediate step between the
parsing engine's (ProcOrder) extraction of information from
an e-mail message and the generation of the valid order.
The intermediate step could be a querying process 112 to
gather additional information from an existing database.

30   The additional information may not have been included in the
outbound e-mail messages but may be needed to generate a
valid order. For example, product codes 112 may be stored
in the database but not included in the outbound e-mail

message. The letters entered by the customer can be mapped to the actual product codes by reference to the tables of the database based upon the source of awareness code.

5   The resulting valid order is a fully-fielded record that has the fields required by the client's order fulfillment system to process an order.

Exception treatment

Exception handling can be treated in different ways depending on the circumstances. For example, an exception
10   might occur when a customer responds from an e-mail client that does not quote the original text of the outbound e-mail message. The inbound e-mail message then has the customer's e-mail address, a subject line that says "yes", and the original subject line from the campaign, but does not have
15   the required information for the shipping address or the footer information. ProcOrder would kick that out as an exception, but the exception handling system would allow a response management representative 46, based on the e-mail address, to confirm, from the database 76, that all of the
20   required information is available. Use of the subject line allows the system to tie back to the appropriate campaign and to figure out who is ordering and what he is ordering. A valid order can be created without further interaction with the customer other than to send him a confirmation that
25   the system has been able to enter a valid order on his behalf.

The system thus recognizes that it is not likely to be possible to automate every interaction with the customer, but it may be possible to complete a dialog with essentially
30   all of the customers from whom inbound e-mail messages are received by automatically identifying messages that will require custom human handling and providing information and

tools that enable the human handlers to complete the
exception transactions in an efficient manner.

Non-order response processing

        Not every inbound e-mail message is an order. Non-
5   order messages include undeliverable bounced messages to ad
hoc customer service responses. Non-order inbound e-mail
messages must be identified by the parsing engine.

        Undeliverable e-mail messages 114 are automatically
separated from the inbound e-mail stream and stored for
10   offline handling by a human response handling professional,
who operates a script on the files of undeliverable
messages. The script classifies them as "soft" and "hard,"
parses e-mail addresses and footer data from the messages,
matches the parsed records to the database, and flags
15   appropriate records as "undeliverable".

        Other non-order messages also are handled manually
as explained earlier.

Vendor creation of e-mail campaigns.

        A campaign creation tool 126 is provided to a vendor
20   to enable simple entry of all information needed to create
an e-mail campaign, including all the parameters, the text
of the messages, and the tables of data needed in the
database. The vendor delivers the campaign electronically
to the transactional e-mail processor which then delivers
25   the e-mail messages, receive the responses, processes all
exceptions, and returns to the fulfillment system the vendor
orders in a proper format.

        A web-based vendor interface 128 enables on-line
viewing by the vendor of the status of all campaigns,
30   including the state of those that are in development and the
results of those that are "live". The information is hosted
by the transactional e-mail processor in part based on the

11

database 76. The interface also gives the vendor a
mechanism to check text and other content into the database.

Alternatively, instead of automatically permitting
the vendor to fully create a finished campaign, the vendor
5   may be enabled to download and check into the database a
proposed campaign. Then an account executive of the e-mail
handler process would review it and work with the vendor to
complete it before it is finally queued for distribution.

Appendices A, B, and C contain more detailed
10   descriptions of aspects of implementations of the invention.
Appendix D contains source code written of an example of the
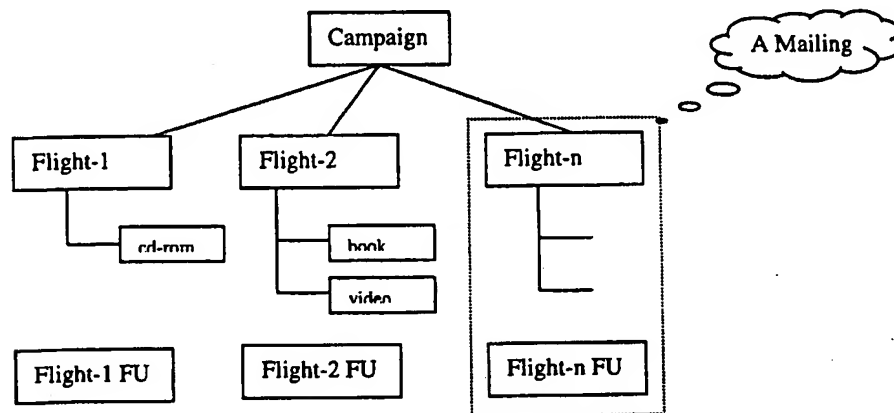ProcOrder process.

Other implementations are within the scope of the
following claims.

15   What is claimed is:

e-mail Protocols, Structures, Definitions & Cycles

MAIL CAMPAIGN

A mail campaign is defined in terms of **Flights**.

A flight contains one or more items that are being promoted but is not restricted to the manner in which the promotion is performed. A flight could be setup to promote a CD-ROM but has variances in the promotion of the item to the targeted mailing. These variances in a flight are called **Tests**.
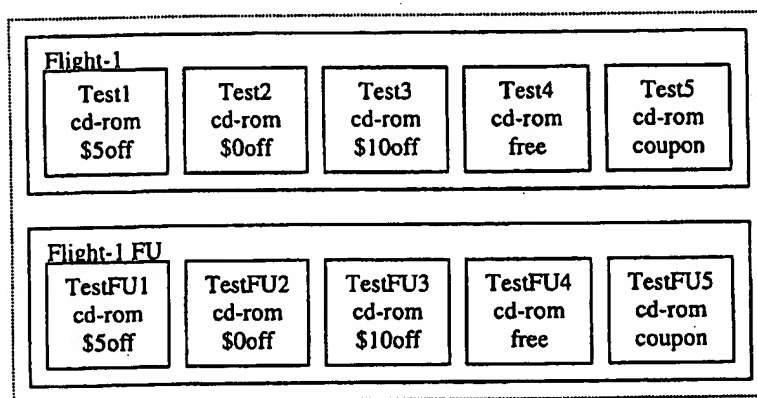
In order to reference the entities above, each are assigned Ids as follows: APPENDIX A

13

A Campaign
- Has a Campaign ID , **CID**
- Has 1 to n flights and flight follow-ups

A Mailing
- Has a Mailing ID, **MID**
- Has a Client Code (SOAC). The mailing is performed on behalf of the client.
- Has a Flight ID, **FID** and a Flight Follow-up ID, **FID**

A Flight
- Has a flight ID, **FID**
- Shares a **SOAC** with its Flight Follow-up
- Contains one or more Test Ids, **TID**
- Has 1 to m items for sale

An Item
- Has an Item ID **IID**
- Has an Item code (determined by the client)

To summarize, the following ID acronyms are used:
- MID   -   Member ID (ID of the person receiving the e-mail)
- CID   -   Campaign ID
- LID   -   maiLing ID
- FID   -   Flight ID
- TID   -   Test ID
- IID   -   Item ID
- SID   -   Style ID (ID of the style of the e-mail message)

MAIL PROTOCOLS

SMTP

The *Simple Mail Transfer Protocol* (SMTP) is described in RFC 821 and is the way that two sites on the Internet exchange mail messages.

The commands are:
- HELO   *domain*
- MAIL   FROM: *username*
- RCPT   TO: *username*
- DATA
- QUIT

Each command is terminated with a CR-LF pair. Replies start with a three-digit response code and continue with text designed to be read by users.

POP3

POP3 is the Post Office Protocol. If the site is always on the Internet, then mail would be sent with an SMTP-sender and received with an SMTP-receiver. However, it may cases it is not possible to maintain a permanent internet connection and in such cases, the Post Office Protocol is used to receive the inbound mail. POP3 allows mail to be stored on machine that is always on the Internet and a receiving host connects to it, asks for any mail and disconnects.

The commands are:
- USER   *username*
- PASS   *password*
- STAT
- LIST   *[message number]*
- RETR   *message number*
- DEL   *message number*
- LAST
- QUIT

THE ANATOMY OF AN OUTBOUND E-MAIL MESSAGE

An outbound email message has a predefined structure. The message is created by combining together a text block with an email address.

The structure of an outbound mail message consists of a Property, Address, Subject, Header, Body, Personal and Token.

Headers
Address
Subject
Salutation
Body
Personal
Token

Structure of an outbound e-mail message

a) Headers
The first few lines of an Outbound Mail message are called the Headers and have a defined format. This information is not normally displayed to the User by a Mail Client application and can only be viewed if the Client permits e.g., in Microsoft Exchange this information can be viewed by listing the properties of a mail message as follows:

```
Received: by mail.kersur.net (mbox peterk)
 (with Cubic Circle's cucipop (v1.31 1998/05/13) Fri Mar 19 20:59:00 1999)
X-From_: aestes@e-dialog.com  Fri Mar 19 13:43:54 1999
Return-Path: <aestes@e-dialog.com>
Received: from montana.e-dialog.com (mail.e-dialog.com [207.31.244.2])
        by mail.kersur.net (8.9.1/8.9.1) with ESMTP id NAA10659
        for <peterk@sytech.com>; Fri, 19 Mar 1999 13:43:53 -0500 (EST)
Received: by MONTANA with Internet Mail Service (5.5.2448.0)
        id <GZZPKKZ2>; Fri, 19 Mar 1999 13:43:57 -0500
Message-ID: <B15DC0490C8AD211BDFD004005A0C2CC47F10B@MONTANA>
From: Anthony  Estes <aestes@e-dialog.com>
To: "'peterk@sytech.com'" <peterk@sytech.com>
Subject: FW: Warning: could not send message for past 4 hours
Date: Fri, 19 Mar 1999 13:43:54 -0500
MIME-Version: 1.0
X-Mailer: Internet Mail Service (5.5.2448.0)
Content-Type: multipart/mixed;
        boundary="----_=_NextPart_000_01BE7238.71C0EB9A"

This message is in MIME format. Since your mail reader does not understand
this format, some or all of this message may not be legible.

------_=_NextPart_000_01BE7238.71C0EB9A
Content-Type: text/plain;
        charset="iso-8859-1"

------_=_NextPart_000_01BE7238.71C0EB9A
Content-Type: application/octet-stream;
        name="ATT00547.TXT"
Content-Disposition: attachment;
        filename="ATT00547.TXT"
```

```
------_=_NextPart_000_01BE7238.71C0EB9A
Content-Type: message/rfc822

Message-ID: <B15DC0490C8AD211BDFD004005A0C2CC47EFF2@MONTANA>
From: Anthony  Estes <aestes@e-dialog.com>
To: "Hillary Gaeth (E-mail)" <HGaeth@engage.com>
Subject: * Thurs, Fri ?
Date: Tue, 9 Mar 1999 08:04:20 -0500
MIME-Version: 1.0
X-Mailer: Internet Mail Service (5.5.2448.0)
Content-Type: text/plain

------_=_NextPart_000_01BE7238.71C0EB9A--
```

A header starts with a *field name* followed by a colon and the field body. The contents of the field body may be rigidly defined or free form.

The following headers are mandatory; that is, there must be a header with each of these names:
- Date
- From, or Sender and From
- To, or CC (carbon copy), BCC (blind carbon copy)

The following headers are optional:
- Return-path
- Received
- Reply-To
- Message-ID
- In-Reply-To
- References
- Keywords
- Subject
- Comments
- Encrypted

Some of these headers are obscure and rarely used. In addition, some mail clients generate their own extra headers. Many such extra header start with the characters "X-" because if extra mail headers are added to the RFC they will never start with these characters.

A header may be split over two lines according to the following rules:
- The split must be at a place where whitespace(blank or tabs) would normally occur; for example, not in the middle of a username or similar field.
- The continuation line must start with a space or tab.

Similary, a header can have extra white spaces almost everywhere except in the middle of a field.

Two headers that are given special attention in the context of this document are those that contain the address and subject.

b) Address
The address is a header that contains the email address of the person receiving the message.

c) Subject

This is a header that describes the subject of the mail. Contained in the subject is a **Subject Token** in the form of two characters. For example, for the mail subject '** A Special Offer for Selected Managers', the **Subject Token** is **

The **Subject Token** is chosen to identify the **Test ID** of the mail message.

d) Salutation

The salutation is a text block in the outbound mail message that is personalized to the person receiving the message. The intention is to provide a personalized greeting and an indication of the sender.

```
From: Harvard Business School Publishing <hbsp@e-care.com>
To: 'esalas@protelsa.com.pe'
Subject: ** A Special Offer for Selected Manager
Date: Tuesday, July 14, 1998 6:47 PM

From the Desk of Laura Winig
Harvard Business School Publishing Corporation
Boston, Massachusetts

Tuesday, July 14, 1998

Ms. Elizabeth Salas
5700 Collins Avenue  Apt  6h
Miami Beach, FL  33140-2308

Dear Ms. Salas:
```

The information in italic is personalized and is combined with the remainder of the message at the time of dispatch.

e) Body

The body contains the main core of the message. The format and layout is fixed and not personalized for a particular test.

```
Simply type "YES" in your reply to this e-mail to take "Virtual Work: Real Results" for a No-
Obligation Test Drive!
=------------------------------------------------------------------=
Do you work with colleagues and clients in multiple locations? Communicate more by email and
conference calls than through meetings? Find your "office" is wherever you are at any given
moment? Then you're working "virtually" and you know that working effectively without proximity
is essential in today's workplace. And you know it's not as easy as it looks.

Here at the publishing arm of Harvard Business School, we've combined extensive research and
real-life examples to create "Virtual Work: Real Results"-a dynamic multimedia program that can
improve your effectiveness working "virtually."
=------------------------------------------------------------------=
Use this engaging tool and you'll understand:

* the dynamics and politics of working virtually-
  and how to handle tough situations when you can't be face-to-face;

* how to most effectively use email, video conferencing,
  voice conferencing-and how to overcome fear of technology;

* ways to build relationships and trust without "human touch"-
```

```
 and how a virtual team can work efficiently and seamlessly.
=------------------------------------------------------------------=
We bring you tricks of the trade from the experts in working virtually.
Then, you gain confidence using these techniques through an interactive case study where you lead
a virtual team through a project. You'll make decisions that determine the success of its
efforts-all in a realistic, but no-risk, environment.

Overcome the isolation and conflicting loyalties that are inherent in working in a virtual
environment-and get ready for success with "Virtual Work: Real Results."
=------------------------------------------------------------------=
Take "Virtual Work: Real Results" for a No-Obligation Test Drive.
=------------------------------------------------------------------=
Simply reply to this email and we'll send you the program with our compliments. We're confident
you'll find you're working more effectively in the virtual world. After 14 days, we'll send you
an invoice for just $295 (single user license).

     But remember, if you're not entirely pleased with the program, simply call us and we'll arrange
                           to pick it up. You will owe nothing.

Sincerely,

Laura Winig
Director
```

In the above example the response with a "YES" is sufficient to indicate the purchase of the single product on offer. In a multi-product offer, the items would be listed and associated with a letter. For this type of mailing, the responder would list the letters in the response.

A mailing may have a follow up 'reminder' flight. This reminder would not go to respondents of the original flight. For example:

```
On Wednesday, July 1, I sent you a special offer on "Virtual Work: Real Results": a new
interactive CD-ROM from Harvard Business School Publishing.

Since I haven't heard back from you, I wanted to send you a reminder Before the offer expires.

If you are simply not interested, I apologize for the intrusion.

=---------  BELOW IS A REPRINT OF THIS SPECIAL OFFER   ---------=
Simply type "YES" in your reply to this e-mail to take "Virtual Work: Real Results" for a No-
Obligation Test Drive!
=------------------------------------------------------------------=
Do you work with colleagues and clients in multiple locations? Communicate more by email and
conference calls than through meetings? Find your "office" is wherever you are at any given
moment? Then you're working "virtually" and you know that working
 . . .
 . . .
 . . .
confident you'll find you're working more effectively in the virtual world. After 14 days, we'll
send you an invoice for just $295 (single user license).

     But remember, if you're not entirely pleased with the program, simply call us and we'll arrange
                           to pick it up. You will owe nothing.

Sincerely,

Laura Winig
Director
```

f) Personal

The personal is added to the outbound message, after the mail body. The purpose of the text block is to request personal details from the respondent. The information requested is presented

in two columns, the first indicating the type of information required and the second as a place for the reply to be entered (in [..]). For example,

```
FIRST NAME:            []
LAST NAME:             []
TITLE:                 []
COMPANY:               []
DEPARTMENT:            []
ADDRESS1:              []
ADDRESS2:              []
ADDRESS3:              []
CITY:                  []
PROVINCE/STATE:        []
POSTAL/ZIP CODE        []
COUNTRY:               []
PHONE:                 []
FAX:                   []
EMAIL:                 []
```

These details are to determine the shipping and billing information.

e) Token
The token follows the Mail Body (or Mail Personal if applicable) and contains information about the mailing and also the addressee e.g., [[878119|2815|1]]

The format of a Mail Token is

$$[[MID \mid TID \mid SID]]$$

where
- MID is an membership ID assigned to the person receiving the outbound message
- TID is the test ID assigned to the outbound message. This would be related to the Subject Token.
- SID is the style ID assigned to the style of the mailing i.e., single product or multi-product.

These three pieces of information uniquely identify the receiver of the outbound message and also the information they received. Hence, the processing of a response is greatly simplified if the reply returns the mail token.

In summary, an outbound message contains generic information that is the same in all the mailings of a test, and also personalized:

Generic	_Personalized_
Subject	Header
Body	Address
	Salutation
	Personal
	Token

THE ANATOMY OF AN INBOUND E-MAIL MESSAGE

An inbound email message has a predefined structure. However, the structure may not be 'structured' sufficiently for it to be automatically processed.

Header
Address
Subject
Response

a) Headers
The first few lines of an Inbound Mail message are Headers and have a defined format. This information is similar in format to that of the outbound message. If the response is produced by replying to an outbound message, (instead of creating it from scratch) then it is probable that the headers of the outbound will be in the inbound mail.

b) Address
The address is a header that contains the email address of the respondent.

c) Subject
This is a header that describes the subject of the mail. This is free-form text and no assumptions can be on its structure. It may be the subject that was used in the outbound mail.

d) Response
The response is a text block containing the message from the respondent. It should not be assumed that the response has any structure since a responder has the freedom to write a reply in "free format" and is not forced to a guideline. This creates a number of problems for the processing of an inbound response since rigid rules cannot be applied.

The points that can be noted about a response are that:
- All responses will contain headers.

- All responses will contain the responders email address.

- All responses will contain a subject. The textual content of the subject cannot be assumed since it can be freely edited and so may not resemble the content of the outbound. If a subject contains a Subject Token then it should correspond to the TID.

- If the response contains the Mail Token then the MID, TID and SID will be available and consequently it will be clear on the approach that should be taken in processing the response i.e., the handling of a single product 'YES' versus a multi-product 'ADG'.
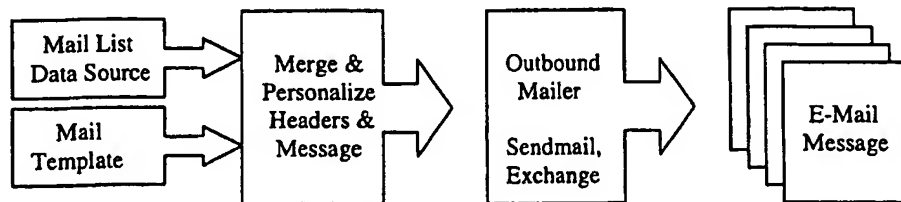
AN E-MAIL LIFE CYCLE

OUTBOUND

a) Create an outbound message

An outbound email message consists of texts block that contains information and one or many questions or choices. The text block of an outbound mail message is sometimes referred to as a **Mail Template.**

b) Deploy the outbound message

The message can be deployed to anyone for whom an e-mail address is available. The address information in a **Mail List** (e.g., database, listserve) is merged with the **Mail Template** and sent. In some cases, the X-Headers are also personalized to reflect the purpose of the outbound mail message.

```
┌──────────────┐                ┌──────────────┐
│ Mail List    │───▶            │ Outbound     │───▶       ┌──────────────┐
│ Data Source  │     ┌────────┐ │ Mailer       │     ┌───▶ │ E-Mail       │
├──────────────┤     │ Merge &│ │              │     │     │ Message      │
│ Mail         │     │Personalize              │     │     └──────────────┘
│ Template     │───▶ │ Headers &              │───▶ │
└──────────────┘     │ Message │ │ Sendmail,    │
                     └────────┘ │ Exchange     │
                                └──────────────┘
```

INBOUND

a) Receive a Reply

The content of a reply can have various forms

- The reply responds to the outbound message e.g., an order.

- The reply requests to unsubscribe from future mailings. The reply contains an indication that the respondent does not wish to receive any further mailings e.g., UNSUB, UNSUBSCRIBE, UNJOIN and REMOVE.

- The reply requests customer service and ad hoc questions.

- The reply indicates a change in personal details or to be added to the mailing list.

There is also the possibility that the outbound message did not reach its final destination and that it bounced back. There are two categories of bounces, hard bounces and soft bounces.

- A hard bounce notification indicates outright failure. For a hard bounce, the subject would contain a message of the following form:
  ```
  Returned mail: Host unknown (Name server: ssoofftteecchh.com: host not found)
  ```

The sender of a hard bounce is usually specific such as:
```
Mail Delivery System [MAILER-DAEMON]
```

A hard bounce can also be detected in the response X-Header:
```
Received: by mail.kersur.net (mbox peterk)
  (with Cubic Circle's cucipop (v1.31 1998/05/13) Fri Mar 19 21:04:39 1999)
X-From_: MAILER-DAEMON  Fri Mar 19 21:04:36 1999
Return-Path: <MAILER-DAEMON>
Received: from localhost (localhost)
    by mail.kersur.net (8.9.1/8.9.1) with internal id VAA20320;
    Fri, 19 Mar 1999 21:04:36 -0500 (EST)
Date: Fri, 19 Mar 1999 21:04:36 -0500 (EST)
From: Mail Delivery Subsystem <MAILER-DAEMON>
Message-Id: <199903210204.VAA20320@mail.kersur.net>
To: <peterk@sytech.com>
MIME-Version: 1.0
Content-Type: multipart/report; report-type=delivery-status;
    boundary="VAA20320.921981876/mail.kersur.net"
Subject: Returned mail: Host unknown (Name server: ssooffttteecchh.com: host not found)
Auto-Submitted: auto-generated (failure)
```

The response body can also contain failure information:
```
The original message was received at Fri, 19 Mar 1999 21:04:35 -0500 (EST)
from dialup111.kersur.net [207.180.95.76]

   ----- The following addresses had permanent fatal errors -----
<JohnSmith@SSOOFFTTTEECCHH.com>

   ----- Transcript of session follows -----
550 <JohnSmith@SSOOFFTTTEECCHH.com>... Host unknown (Name server: ssooffttteecchh.com: host not
found)
```

- Softbounces can be of three types:
    ○ *NonDeliveryNotification* occurs when a given message has not been delivered yet but will continue to try and deliver for a further specified period of time. This state can be detected in the response subject:
    ```
    FW: Warning: could not send message for past 4 hours
    ```

    The response can also be detected in the response body:
    ```
    ••••••••••••••••••••••••••••••••••••••••••••••
    ••      THIS IS A WARNING MESSAGE ONLY       ••
    ••  YOU DO NOT NEED TO RESEND YOUR MESSAGE   ••
    ••••••••••••••••••••••••••••••••••••••••••••••
    The original message was received at Tue, 9 Mar 1999 08:00:18 -0500 (EST)
    from mail.e-dialog.com [207.31.244.2]

       ----- The following addresses had transient non-fatal errors -----
    hgaeth@andexc01.cmgi.com
        (expanded from: <HGaeth@engage.com>)

       ----- Transcript of session follows -----
    hgaeth@andexc01.cmgi.com... Deferred: Connection refused by
    andexc01.cmgi.com.
    Warning: message still undelivered after 4 hours
    Will keep trying until message is 3 days old
    ```

    ○ *AutoResponders* are notifications which actually indicate delivery but are sent by mail agents to indicate that the user will not be able to respond immediately (possibly on vacation) but the sender should expect a response when they return.
    ○ *Unknown*

b) Process the reply

Automatic processing of an e-mail response is defined as the ability to determine accurately the requirements of the reply by using text inspection and search rules.

To automatically process an e-mail response, there are a number of criteria that need to be satisfied by the content of the reply. The main three criteria are:

ci)   Who is the Respondent
cii) What outbound message is the response to
ciii) What does the Respondent require

The exception to the above are hard and soft bounces that are identified by other e-mail properties.

Since all legitimate mail responses will contain the e-mail address of the Sender this can be regarded as a base information for all responses. This information fulfills criteria ci) but alone is not sufficient for a response to be processed automatically.

Criteria cii) can be satisfied by the Subject Token, provided the respondent has not altered it. Note that the Style ID is implied if criteria cii) is satisfied.

Another source of information is the Mail Token. For a reply that contains a Mail Token, both criteria ci) and cii) can be determined.

But how is criteria ciii) satisfied? There is no simple solution to this question since the answer lies in the body of the reply, which is in 'free form'. The only method available that can determine the requirements of the respondent is to parse the email reply based on the SID.

c) Produce Reports

After determining the requirement of the inbound e-mail, database entries and reports can be produced.
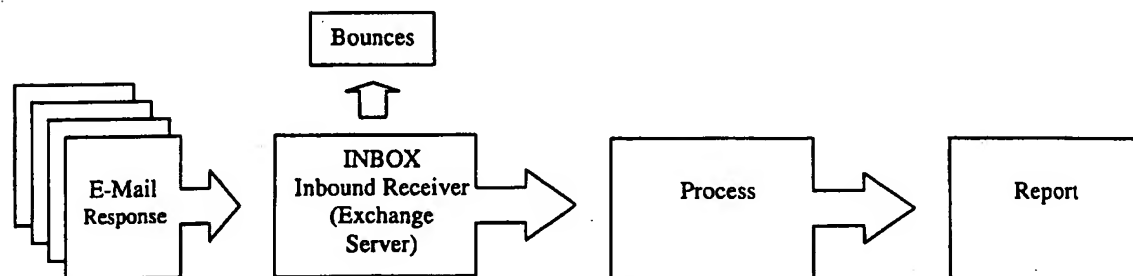
Design Specification for
QuickReply

QUICKREPLY

The QuickReply application is used to process e-mail responses to mailing campaigns. The building and sending of the e-mail messages is not within the scope of QuickReply.

The test within a campaign's flight provides the outbound e-mail message. The message will contain both generic and personalized information and will normally require a response from the person the message is sent to.

Responses to a test are received in the Inbox folder of the mailbox. The design objective of QuickReply is to read the responses from the Inbox, and decide from its content the requirements of the response. On determining (or not determining) the requirement, the response is filed to pre-configured sub-folders and the appropriate tracking database is up-dated.

QuickReply will generate diagnostic information that will support the reasoning on how it reached its decision.

The main data storage area of QuickReply is Microsoft Access relational database. Appropriate records will be added to this database as the response processing is performed. In addition, it is likely there will be responses that cannot be automatically processed and, in such cases, Customer Service will use QuickReply in manual mode. Manual mode will also add records to the appropriate tables in the database.

The information in the database will be used to generate reports.

APPENDIX B

MAILBOX ORGANIZATION

Mailings are performed on behalf of a Client. Consequently, all e-mail transfers for a particular Client will be performed within the Clients mailbox.

The organization of the Clients mailbox includes an Inbox for the responses and a pre-configured number of sub-folders where the responses are filed.

Mailbox - <Client Reference e.g., HBSP)
    Campaign CID
        Flight FID
            AC
                YYMMDDhh
            AddChange
                YYMMDDhh
            BouncesHard
                YYMMDDhh
            BouncesSoft
                YYMMDDhh
            Master
            Order
                YYMMDDhh
            Unclear
                YYMMDDhh
            Unsubscribe
                YYMMDDhh
    Inbox
        YYMMDDhh

In the above, YYMMDDhh is a directory named after the year, month, day and hour. This format reflects the date and time the entries were placed in that folder.

Each test flight will have a number of responses associated with it. When QuickReply categorizes a response, it will be automatically moved from the Inbox to the appropriate sub-folder.

-	AC	Responses for customer service to process manually
-	AddChange	Responses that contain personal detail changes
-	BouncesHard	Hard bounces
-	BouncesSoft	Soft bounces
-	Master	Master version of the outbound mail
-	Order	Responses with an order
-	Unclear	Responses that are unclear as to how they should be processed
-	Unsubscribe	Response to unsubscribe

E-MAIL STYLES

The Style ID (SID) of a response indicates the approach that should be taken by QuickReply during processing.

The SID is a two-digit numerical value

XY

where:
- X=1 indicates the outbound Personal block is included in the outbound message.
- Y=message layout style described below.

The following two layout styles are supported:

a) Single Product Offer – layout style=1
This style offers one product. The respondent is asked to reply "YES" if they are interested.

b) Multi-Product Offer – layout style=2
The mailing information is based on information in a relational database. This style offers a number of products the respondent can choose from. For example, the outbound message could contain:

```
Your choices (detailed below) are:

A:    The work of Leadership Audio
B:    Leading Your People Audio
C:    Leading Change Successfully Audio
D:    Overcoming Resistance to Change Audio
E:    Gaining Competitive Advantage Audio
```

The respondent makes a choice and responds by stating their requirement in the free format e.g., "please send me ABE".

The number of items can vary as well as the choice symbol i.e., 1,2,3, could be used instead of A,B,C.

PROCESSING INBOUND MAIL AND REPORTING

The general processing and reporting function is described below. The general flow in processing responses is to determine the membership ID (MID), then the test ID (TID) and style ID (SID) and then perform an examination of the response to determine the requirement.

The processing criteria can be summarized as follows:
ci)   who is the Respondent?
cii) what is the response to?
ciii) what does the Respondent require?

All inbound responses are targeted to the Inbox of the Client's Mailbox. On a periodic basis, the responses in the Inbox will be processed.
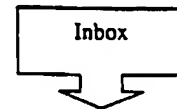
ci) Who is the respondent? – Determine the MID
The are several ways in which the Membership Information can be determined. The importance of the MID is that could provide a handle to the Members personal information

a) Extract the Response Address from the response. Determine if a unique record exists for the Address. If a unique address exists then the MID is determined. If the record is not unique then examine the mail token.
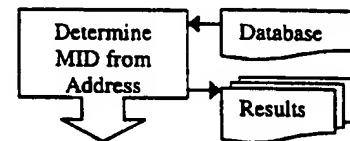
b) If the Mail Token exists then the MID is determined. Use the MID to confirm that the respondents e-mail address is the same as that in the database. If the addresses are different then report that this was the case but continue to process.
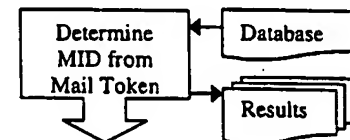
c) If the SID indicates that the Personal block was included in the outbound, the extract the Member details from the Mail Personal. The extraction assumes that the response information is in square brackets '[]'. The field definitions are defined in the database together with an indication of those that have to be completed by the responder. If Mail Personal is not complete (the fields are indicated in the database), move the response to Unclear and report.
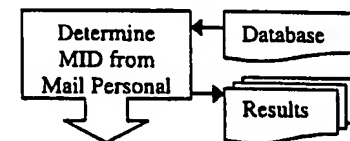
d) If the MID cannot be determined, move the response to Unclear and report.

cii) What is the response to? – Determine the TID and SID
The next stage in response processing is to determine what the response is to. This can be determined from the Test ID.

a) If the Mail Token exists then extract the TID and SID. If the Mail token is absent, report but continue the determination of the TID.
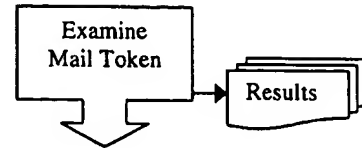
b) Determine the TID from the database based on the Mail Subject of the response. If the look-up is successful, then the SID will also be determined.
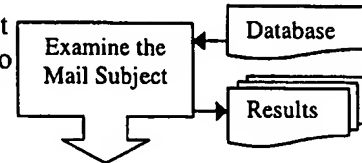
c) At this stage the TID and SID should be known. If not, move the response to Unclear and report.

ciii) What does the Respondent require?

With the MID, TID and SID determined, the response message(s) are still of no value until it is established what the respondent requires. The obvious approach to this problem is to compare the response with the original and try to make some sense of the differences.

a) If the Mail Subject did not contain the Subject Token, examine for words that appear in the word action table. Any bounced responses are moved to the Bounce sub-folders. Any unsubscribe requests are moved to the Unsub sub-folder.

```
Examine the          Database
Subject for
'words'              Results
```
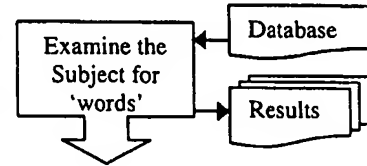
b) If the SID indicates a single item promotion, examine the Mail Subject for words that appear in the word action table. Any order responses are moved to the Orders sub-folders.

```
Examine the          Database
Subject for
'order'              Results
```
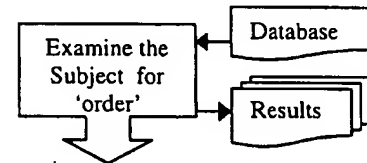
c) Examine the response for words that appear in the word action table. Any bounced responses are moved to the Bounce sub-folders. Any unsubscribe requests are moved to the Unsub sub-folder.

```
Examine the          Database
Response for
'words'              Results
```
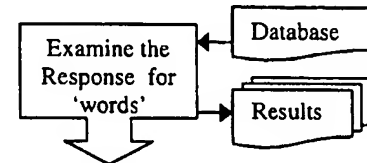
d) Remove all the text of the outbound message from the response.

```
Normalize            Database
the Response
```
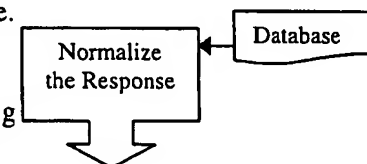
e) Normalize the response. Remove characters from the beginning of each line. The list of characters to be removed are in the word replacement table with ModeID=LineStart.

f) Replace phases from the normalized response. The phases that are replaced are given in the word replacement table with ModeID=PhaseReplace.

```
Re-normalize         Database
the Response
                     Results
```
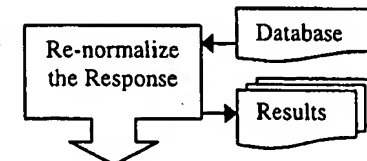
g) If the SID indicates that there is only a single item on promotion, examine the normalized response for words that indicate an order of a single product item. The words are in the word action table with the ActionID=SingleOrder. If order

```
Examine              Word list
Response for
'single'             Results
```
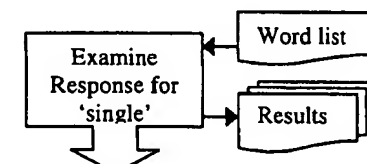
h) If the SID indicates that there are multi-items on promotion, examine the normalized response for phrases that indicate an order of multi-items. The words are in the word action table with the ActionID=MultiOrder. In the case of multi-products, the

```
Examine              Word list
Response for
'multi'              Results
```

30
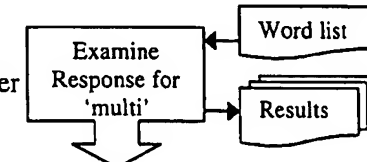
information the StartOption and EndOption from the bindings table are used to verify that item(s) ordered are in the correct choice range.

i) Any order requests are moved to the Orders sub-folder.

j) If the purpose of the response cannot be determined, then move the response to the Unclear sub-folder.

DATABASE ORGANIZATION

QuickReply uses two types of databases that are stored in Microsoft Access. The first database is part of the QuickReply application and contains high-level information on the campaigns, flights and tests. The second database is specific to a particular Client Test.

a) QuickReply Database
The QuickReply database contains the information that binds campaigns, flights and tests.

tblTestBindings	**- contain Test bindings**
Test ID	**unique**
Client ID	
Client Test Database	database name used for the test information
Subject	subject text of the outbound test
Style ID	the style of the outbound message
StartOption	character of the first item in the test
EndOption	character of the last item in the test
Campaign ID	
Mailing ID	
Flight ID	

tblOutboundPersonal	**- contains personal details that appear in a test**
Test ID	**unique**
Question	prompt displayed in personal block e.g., First Name
AnswerLen	the maximum length of the answer
AnswerReq?	flag to indicate that an answer must be given

tblWordActions	**- contains words that convey an action**
Word	**unique**
Action ID	action id for this word e.g., 'bounce'

tblWordReplacement	**- contains phrases and their replacement**
Phase	**unique**
Replacement	replacement to the phrase
Mode ID	type of replacement

b) Client Test Database

tblMembers	**- contains member details**
Member ID	**unique**
Email	
Prefix	
First	
Middle	

Last
Suffix
Title
Company
BillAddress1   ShipAddress1
BillAddress2   ShipAddress2
BillAddress3   ShipAddress3
BillCity   ShipCity
BillState   ShipState
BillZip   ShipZip
BillCountry   ShipCountry
Tel
Ext
Fax

EDRMA

Introduction

This document is intended to provide a schematic synopsis of E-Dialog Response Management Architecture (EDRMA).

Thereby it is hoped that the design of the Verbind E-Mail Channel Server (ECS) can best accommodate EDRMA's input requirements, and so EDRMA's output requirements may be adjusted to match those of Verbind LifeTime's architecture (VLTA)

> "First, some acronyms…" - *Anonymous*

Relevant Modules - Top [0.1]
Where EDRMA Fits w/ECS

EDRMA

OUTBOUND

MailFile
Outputs from VLTA II
Other dataSources

Template

Merger
(ECS-MER)

Outbound Mailer
(Sendmail, Exch-Connector)
(ECS-MAI)

Mailing
Instances

Bold modules indicate ECS modules

INBOUND

- Hard Undels
- Soft Undels
- Notifications

Noise We
Must Count

Response
Instances

Web Profile
Responses

Triager
(ECS-
TRG)

Web Thingy
(ECS-TRG-
WWW)

ID Resolver
(ECS-TRG-
IDR)

Inbound Receiver
(Exchange Server)

EDRMA

Client Mail
Stores

EDRMA   Data Entity Relationships [0.5]   Model

Campaign
- has 1 CID
- has 1:N Flights (FIDs)
- has 1 template class (TCID)
  - eNews
  - QuickReply - single
  - QuickReply - multi
- Approved Response Templates (template ID)

Mail Key Codes
- [[Cust|DI|List|DIF|DIF|TCID] <now>
- [[MIID|FID|TCID]] <future>
- **MIID** -> global mail instance ID
- placement: X-Header, Subject, Footer

Mailing (e.g., F1 & F1-FU)
- has 1 MID
- have 2 FIDs, one for F, FU
- has one SOAC (client code)

Any Flight has 1:N items for sale
- has one FID
- has one TCID
- (typ) shares a SOAC with its FU
- Any Item
  - has 1 InstanceID (our code)
  - has 1 ItemID (client code)

together, a "Mailing"

Campaign

Flight-1   Flight-2   Flight-N

Item 1: "book 5"
Item 2: "cd-rom 2"
Item 3: "subscription 7"

Flight-1 FU   Flight-2 FU   Flight-N FU

master-1   master-2   master-3

master-1 FU   master-2 FU   master-3 FU

Template Instances

EDRMA   Mailbox

Canonical Data Structures - Exchange Server Data Store [0.2]

Mailbox - <ClientFullName> *(not alias: "hbsp")*

<CampaignName>[<CID>]

- ApprovedResponseTemplates <templateID>
- <FlightName>[<FID>]
  - AC *(i.e., "Advocacy Care")*
    - + <date: YYMMDDhhmm>
  - ADDCHANGE
    - + <date: YYMMDDhhmm>
  - HARDBOUNCES *(i.e., hard undelivereds... "jsmith@foo.com is not a valid addressee")*
    - + <date: YYMMDDhhmm>
  - INBOX (monolithic, this Mailbox)
  - MASTER (contains master template, this FID)
  - ORDERS
    - + <date: YYMMDDhhmm>
      - » [[custID|ListID|Flight|DITCID]] << footer tags (x-header, subject)
  - SOFTBOUNCES
    - + DELIVERYNOTIFICATIONS
      - » <date: YYMMDDhhmm>
    - + AUTORESPONDERS
      - » <date: YYMMDDhhmm>
    - + UNKNOWN
      - » <date: YYMMDDhhmm>
  - UNCLEAR
    - + <date: YYMMDDhhmm>
  - UNSUBS
    - + <date: YYMMDDhhmm>

EDRMA

Selected Applications

VBA.PKinboxInspector
interface: GUI / VBA
data: MAPI
inbox sorting, folder management application

VBA.PKresponseProcessor
interface: GUI / VBA
data: MAPI
response review and report preparation application

PL.AEprocOrder()
interface: commandline
data: ADO 2.0
process raw "order e-mails" by TCID
uses cf file rulesets for document preprocessing and data element parsing tied to CID; other rules, hardcoded
generates:
    Acceptable output for review, annotated
    Exceptions, annotated
    Truncated BODYs, annotated
    Raw fields output , annotated
    Raw fields exceptions , annotated
    Rule-eval log (exhaustive)

PL.AEprocBatPrep()
interface: commandline
data: ADO 2.0
takes selected output from PL.AEprocOrder() and transforms to a batch transfer specification via a field map and transform rules cf

PL.AEprocUpdateBatVerbInd()
TBD

PL.AEscrubNormalCanon()
<< not used on the response side >>
interface: commandline
data: file handles
scrubber, normalizer, canonicalizer
also generates scrambled (non-predictable) row Ids and flags AOLs

EDRMA

Process Stages [0.5]

Preliminary ECS hooks anticipated

Preprocess Stage
- inspect Inbox using VBA.PKinboxInspector
  - auto-creates folder structure (previous slide)
  - facilitates auto-sort of Items into ORDERS, UNSUBS, UNDELS, HARDS, SOFTS, ADDCHANGE, etc.
  - facilitates manu-sort of Items into AC, and exception
- report preparation by TCID via AEprocOrder()
  - produces: OUT_report, EXC_report && EXC_BODY diagnostic report

Processing Stage using VBA.PKresponseProcessor
- for each EXC_
  - inspect, correct, reconcile (via Mailing Table lookups) and commit
- for each OUT_
  - inspect and commit to report (or not)
- Acceptable UNION of EXC_ && OUT_ -> Reporting, Update Stages { AEprocBatPrep() || AEprocUpdateBatVerbind() }
- Hard exceptions are tagged as such and become Followup RFC's to get additional (critical information)
- FUP (FUPID -> RFC) tags affect routing; tie-back to HARD exception row in this EXC_ report for closure

Response Follow-up Confirmation Stage
- for each non-BOUNCE (ORDERS, UNSUBS, etc.), respond to respondents with an appropriate "confirmation of receipt/action taken" message
- using relevant response template store (templateID)
- this class of response likewise tagged for routing (FUPID -> confirm)

Reporting Stage
- produce order report
  - deliver via fax
  - post to client private web application
  - deliver as formatted batch

Update Stage
- deliver formatted batch to {Verbind, Database} -> sp_???, SQL Executive

Page 1

APPENDIX D

```
   F:\Development\PR...\PR...hbsp_extractOrder.WARNIN SpreadWHAT.pl
   Printed at 19:48 on 06 Feb 1999

 1 # procorder.pl (c) 1998,1999 E-Dialog, Inc. All Rights Reserved.
 2 #
 3 #
 4 # PURPOSE DISCUSSION
 5 #
 6 # program accepts a raw email stream from MS Exchange Server via export to ADO-compliant datastore
 7 # then uses email body preprocessing and extraction rules - tailored per campaign in ./cf/ - to construct order reports.
 8 #
 9 # cf files are campaign optimized:
10 #
11 #    mailingDBLayout.cf                    -> field mappings for a campaign's master mailing table
12 #    messageBodyDataLayout.cf              -> DSNs, field mappings, critical rule designations, order-designation bounds, synonym table
13 #    messageBodyIndicateaddresschange.cf   -> rulesets for indicating ADDRESS CHANGES
14 #    messageBodyIndicateBuyAll.cf          -> rulesets for indicating BUY ALL state
15 #    messageBodyPreprocRules.cf            -> body text preprocessor configuration and rulesets
16 #
17 # program produces diagnostic and order-entry bound output files and logs outcomes of all ruleset evaluations leading to
18 # preparation of those output files, which include:
19 #    <$ARGV[1]>           -> summary order report file (designed for data entry, faxing, emailing to fulfillment)
20 #    EXC_<...>            -> (companion) summary exception report (same format as s. order report with annotations to indicate rule failures for man
21 # ual review)
22 #    OUT_<...>           -> detailed order stream file (all fields discretely mapped and normalized for input into batch postprocessor)
23 #    EXC_DETAIL_<...>    -> (companion) ..exceptions..discrete fields etc.
24 #    EXC_BODY_<...>      -> annotated diagnostic report containing all message bodies - sans original message - for aid in resolving exceptions man
25 # ually
26 # output files are (default) tab-delimited text
27 #
28 # PROGRAM REQUIRES
29 #
30 #    Perl 5.003 interpreter later compiled for WIN32
31 #    MS ADO components 1.5 (2.0+ recommended)
32 #
33 # RELEASE HISTORY
34 # 981020AE: 0.5     first usage; to extract caroll0 order reports
35 # 981204AE: 0.9     upgrade to handle caroll2, multiple item selections
36 # 981208AE: 1.1     improve flagging
37 # 981211AE: 1.5     adapt to extract IDEAS@WORK add/changes
38 # 981212AE: 2.3.1   fix bugs in EXC_BODY reporting / enhance it
39 # 981222AE: 2.4     add cfPATH support
40 # 981223AE: 2.4.1   cfPATH logging; fix synonym (SUBJECT TOKEN) lookup <trailing \s*>
41 # 981223AE: 2.5     reorder reporting fields
42 # 981224AE: 2.5.1   reorder reporting fields
43 # 981222SAE: 2.5.2  update SHIPPING_ADDRESS fields to reflect presence of T,B & S diagnostic fields
44 # 981226AE: 2.6     code review/dox; update cf to require MAILINGID
45 # 990106AE: 2.6.5   standardize on LOG_ID for use in address change
46 # 990129ae: 2.7     add support for separate billing, shipping addresses
47 # 990201ae: 3.0     add record-recovery via ADO -> mailing database lookups
48 # 990201ae: 3.0.1   record-update ADO -> mailing database
49 # 990202ae: 3.0.2   fix leading digits bug in getDateTimestamp ftn
50 # 990203ae: 3.1     fix MDB query bugs; introduce detection of SELECT && UPDATE SUCCESS; verify decision-tree based UPDATES
51 # 990205ae: 3.1.1   fix address3 bug; provide regression compatibility for /ADDCHANGE
52 # 990206ae: 3.1.2   throw .noTok as a HARD_EXC
53 #
54
55 ############### INCL ###############
56
57 use win32::OLE;
58
59 ############### ENV ###############
60
61 $GrequiredARG  = 4;
62 $Grevision     = '3.1.2';
63 if(scalar(@ARGV) < $GrequiredARG) { die "($Grevision) usage: procorder.pl [userName] [reportName] [cfPATH <campaignMnemonic>] [ <buyToken> | /multi | /add
64 Change ]\n"; }
65
66 $Guser         = $ARGV[0];
```

```
F:\Development\PRJ\PRJ_hbsp_extractOrder\WORKING\extractOrder.pl
Printed at 19:48 on 06 Feb 1999

67   $reportFile                 = $ARGV[1];
68   $ccfpath                    = $ARGV[2];
69   $subj_buyToken              = $ARGV[3];
70   $excReportFile              = 'EXC_'.$reportFile;
71   $excBODYreportFile          = 'EXC_BODY_'.$reportFile;
72   $outFile                    = 'OUT_'.$reportFile;
73   $exceptionsFile             = 'EXC_DETAIL_'.$reportFile;
74
75   if($ARGV[4] == 3} { $GDEBUG = 1; }  else {$GDEBUG = 0;}
76   if($ARGV[4] == 2} { $LGDEBUG_FTN = 1; } else {$LGDEBUG_FTN = 0; }
77
78   ################## DEFN ##################
79
80   ## MISC GLOBALS ##
81
82   $platform              = 'WIN';
83   $eventType             = 'INFO';
84   $basePath              = '.';
85   $GfieldDelimiter       = "\t";
86   $Gspace                = ' ';
87   $Gnewline              = "\n";
88   $Gcomma                = ',';
89   $Gpipe                 = '|';
90   $eventRequiresFUP      = 'MULTI';
91   $GmultiItemOrderSwitch = '';
92   $GaddChangeSwitch      = 'ADDCHANGE';
93
94   ############ OUTPUT FILE LAYOUTS ############
95
96   ## DO NOT change alpha ordering of lowercase field names (e_ ... t_)
97   ## This is vital in the BODY field layout defn file since the rules reference the letter prefixes
98
99   ## reportFile (note: EXC_reportFileName also written from this template)
100
101  %reportFileRecord = (
102
103      a_LID                   => '_e_$thisLogkey',
104      ab_LINE                 => '<fill-in>',
105      b_RID                   => '_e_$thisRecordID',
106      c_THIS_SUBJECT          => '_e_$thisSubject',
107      d_THIS_BODY_BOTTOM      => '_e_$thisBodyBottom',
108      e_THIS_BODY_TOP         => '_e_$thisBodyTop',
109      f_FUP_REQUIRED          => '_e_$eventRequiresFUP',
110      ib_LOCATION_BUY_TOKEN   => '_e_$thisBuyStatus',
111      j_REVIEW_ACTION_TAKEN   => '<review>',
112      za_SELECTED_ITEMS       => '_e_$thisitemsselectedstack',
113      zd_PRIORITY             => '_e_$thisMailingID',
114      zf_CUSTNO               => '_e_($thisCustNo==-1||$thisCustNo == 0)?"":$thisCustNo',
115      zi_LISTID               => '_e_($thisListID==-1)?"":$thisListID',
116      zm_FROM_ADDRESS         => '_e_$thisFromAddress',
117      zp_SPECIAL_INSTRUCTIONS => '<none>',
118      zr_PHONE                => '_e_$results{\'r_bphone\'}',
119      zx_BILLING_ADDRESS      => '_e_$thisBillingAddressBlock',
120      zy_SHIPPING_ADDRESS     => '_e_$thisShippingAddressBlock',
121      zz_MDB_BILLING_ADDRESS  => '_e_$thisMDBbillingAddressBlock',
122
123  );
124
125  ## outFile
126
127  %outFileRecord = (
128
129      a_LID                   => '_e_$thisLogkey',
130      aa_RID                  => '_e_$thisRecordID',
131      ab_THIS_SUBJECT         => '_e_$thisSubject',
132      ac_THIS_BODY_BOTTOM     => '_e_$thisBodyBottom',
133      ad_THIS_BODY_TOP        => '_e_$thisBodyTop',
134      b_LISTID                => '_e_$thisListID',
135      bb_CUSTNO               => '_e_$thisCustNo',
```

```perl
E:\Development\PR\PRJ_hbsp_extractOrder-WORKING\procOrder.pl
Printed at 19:48 on 06 Feb 1999

136      C_PRIORITY            => '_e_$thismailingID',
137      e_bfirst              => '_e_$results['e_bfirst']',
138      f_bmiddle             => '_e_$results['f_bmiddle']',
139      g_blast               => '_e_$results['g_blast']',
140      h_btitle              => '_e_$results['h_btitle']',
141      i_bcompany            => '_e_$results['i_bcompany']',
142      j_bdepartment         => '_e_$results['j_bdepartment']',
143      k_baddress1           => '_e_$results['k_baddress1']',
144      l_baddress2           => '_e_$results['l_baddress2']',
145      m_baddress3           => '_e_$results['m_baddress3']',
146      n_bcity               => '_e_$results['n_bcity']',
147      o_bstate              => '_e_$results['o_bstate']',
148      p_bpostal             => '_e_$results['p_bpostal']',
149      q_bcountry            => '_e_$results['q_bcountry']',
150      r_bphone              => '_e_$results['r_bphone']',
151      s_bfax                => '_e_$results['s_bfax']',
152      t_bemail              => '_e_$results['t_bemail']',
153
154      ue_sfirst             => '_e_$results['ue_sfirst']',
155      uf_smiddle            => '_e_$results['uf_smiddle']',
156      ug_slast              => '_e_$results['ug_slast']',
157      uh_stitle             => '_e_$results['uh_stitle']',
158      ui_scompany           => '_e_$results['ui_scompany']',
159      uj_sdepartment        => '_e_$results['uj_sdepartment']',
160      uk_saddress1          => '_e_$results['uk_saddress1']',
161      ul_saddress2          => '_e_$results['ul_saddress2']',
162      um_saddress3          => '_e_$results['um_saddress3']',
163      un_scity              => '_e_$results['un_scity']',
164      uo_sstate             => '_e_$results['uo_sstate']',
165      up_spostal            => '_e_$results['up_spostal']',
166      uq_scountry           => '_e_$results['uq_scountry']',
167      ur_sphone             => '_e_$results['ur_sphone']',
168      us_sfax               => '_e_$results['us_sfax']',
169      ut_semail             => '_e_$results['ut_semail']',
170
171      ux_FROM_ADDRESS       => '_e_$thisFromAddress',
172      uy_LOCATION_BUY_TOKEN => '_e_$thisBuyStatus',
173      v_SUBJECT             => '_e_$thissubject',
174      z_FUP_REQUIRED        => '_e_$eventRequiresFUP',
175      zz_REVIEW_TYPE        => '_e_$EXCEPTION_TYPE',
176
177      );
178
179  ## exceptionsFile - BODY only
180
181      %GexceptionsBODYFileRecord = (
182
183      a_LID            => '_e_("
184      \n>>'.$thislogkey."<<")',
185      aa_REVIEW_TYPE   => '_e_$EXCEPTION_TYPE',
186      ab_EXCEPT_FLAGS  => '_e_$eventRequiresFUP',
187      b_RID            => '_e_("<<".$thisRecordID.">>")',
188      u_FROM_ADDRESS   => '_e_("<<".$thisFromAddress."||")',
         V_SUBJECT        => '_e_("||".$thissubject."||\n=====
189      w_BODY_BELOW     => '_e_$thisFullNumberedBody',      # full body
190
191      ===================")',
192
193      );
194
195  ## activityLogFile
196
197      %GactivityLogRecord = (
198      a_LID         => '_e_$thislogkey',
199      b_user        => '_e_$Guser.':'.$GccPath',
200      c_dateStamp   => '_e_&getDateStamp('WIN',$LGDEBUG_FTN)',
201      e_RID         => '_e_$thisRecordID',
202      f_clientName  => '_e_$GclientName',
         g_eventType   => '_e_$eventType',
```

```
     E:\_Development\PRJ\PRJ_hbsp_extractOrder-WORKING\probOrder.pl
     Printed at 19:48 on 06 Feb 1999

203        h_eventDesc              => '_e_$eventDesc',
204        z_FUP_REQUIRED           => '_e_$eventRequiresFUP',
205     );
206
207
208    ############### DATA #####################
209    &getkeypos_outfile(\%Gsoutfilerecord);            # identify the fieldwise position of fields in the outfile for canonicalization (use exc file as model)
210    ##print "--- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
211
212    $GDSN                         = 'dsnorderProcessing';
213
214
215    if ($Gplatform ne 'WIN') {$relativePath = '/logs/'} else { $relativePath = '\\logs\\';}
216    $Gactivitylogfilename         = $basePath.$relativePath.'activitylog.tab';
217    $GactivityLogkeysFileName      = $basePath.$relativePath.'activitylogkeys.tab';
218
219    if ($Gplatform ne 'WIN') {$relativePath = '/db/'} else { $relativePath = '\\db\\';}
220    $GexReportDBkeysFileName       = $basePath.$relativePath.'excReportDBkeys.tab';
221    $GexceptionsDBkeysFileName     = $basePath.$relativePath.'excDetailDBkeys.tab';
222    $GexceptionsBODYDBkeysFileName = $basePath.$relativePath.'excBodyDBkeys.tab';
223    $GoutFileDBkeysFileName        = $basePath.$relativePath.'outFileDBkeys.tab';
224    $GreportFileDBkeysFileName     = $basePath.$relativePath.'reportFileDBkeys.tab';
225
226    if ($Gplatform ne 'WIN') {$relativePath = 'cf'.'/'.$GcfPath} else { $relativePath = '\\cf\\'.'.'$GcfPath\\';}
227    $GbodyDataLayoutFileName       = $basePath.$relativePath.'messageBodyDataLayout.cf';
228    $GbodyPreProcRulesFileName     = $basePath.$relativePath.'messageBodyPreProcRules.cf';
229    $GbodyBuyAllFileName           = $basePath.$relativePath.'messageBodyIndicateBuyAll.cf';
230    $GbodyAddressChangeFileName    = $basePath.$relativePath.'messageBodyIndicateAddresschange.cf';
231    $GmailingDBLayoutFileName      = $basePath.$relativePath.'mailingDBLayout.cf';
232
233    ##################################
234    ############ MAIN ################
235    ##################################
236
237    $MODE            = '';
238    $MULTI_BUY_ON    = 0;
239    $ADD_CHANGE_ON   = 0;
240    $Gsubj_buyToken  = uc($Gsubj_buyToken);
241
242    $MODE = "<TOKEN = |$Gsubj_buyToken|>";
243    if($Gsubj_buyToken =~ /\MULTI/)     { $MULTI_BUY_ON = 1; $MODE .= 'MULTI BUY ON>';}
244    if($Gsubj_buyToken =~ /\ADDCHANGE/) { $ADD_CHANGE_ON = 1; $MODE .= 'ADDCHANGE ON>'; }
245
246    print "[procorder R.$Grevision] $MODE\n".($GDEBUG?"\n<DEBUG: $GDEBUG;$LGDEBUG_FTN>\n\n":"\n");
247
248    ## open cf files
249
250    print "*** opening: $GbodyPreProcRulesFileName...\n";
251        @GPreProcRules               = &loadFile_SCALAR($GbodyPreProcRulesFileName);
252
253    print "*** opening: $GbodyBuyAllFileName...\n";
254        @GBuyAllIndicators           = &loadFile_SCALAR($GbodyBuyAllFileName);
255
256    print "*** opening: $GbodyAddressChangeFileName...\n";
257        @GaddressChangeIndicators    = &loadFile_SCALAR($GbodyAddressChangeFileName);
258
259    print "*** opening: $GbodyDataLayoutFileName...\n";
260        %LAYOUT                      = &HASHloadFieldMap($GbodyDataLayoutFileName,$LGDEBUG_FTN);
261
262    unless($ADD_CHANGE_ON) {
263        print "*** opening: $GmailingDBLayoutFileName...\n";
264        %MDBLAYOUT = &HASHloadFieldMap($GmailingDBLayoutFileName,$LGDEBUG_FTN);
```

43

```perl
F:\Development\PRJ\PRJ_hbsp_extractOrder\WORKIN\Gproc\Order.pl
Printed at 19:48 on 06 Feb 1999

272  }
273
274  ## open I/O
275
276  if (!open(FH_log, ">>$gactivityLogFilename"))         {$eventType = 'ABORT'; $eventDesc = "MAIN::Cannot open GactivityLogFilename: \[$filename]"; &loge
     vent(\*FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);}
277  if (!open(FH_out, ">$goutFile"))                      {$eventType = 'FAIL'; $eventDesc = "MAIN::Cannot open GoutFile: \[$goutFile]"; &logEvent(\*FH_log
     ,$eventType, $eventDesc, $guser, $LGDEBUG_FTN);}
278  if (!open(FH_report, ">$greportFile"))                {$eventType = 'FAIL'; $eventDesc = "MAIN::Cannot open GreportFile: \[$greportFile]"; &logEve
     FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);}
279  if (!open(FH_excreport, ">$gexcReportFile"))          {$eventType = 'FAIL'; $eventDesc = "MAIN::Cannot open excReportFile: \[$gexcReportFile]";
     nt(\*FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);}
280  if (!open(FH_excBODYreport, ">$gexcBODYReportFile"))  {$eventType = 'FAIL'; $eventDesc = "MAIN::Cannot open excBODYReportFile: \[$gexcBODYReportFile]";
     &logEvent(\*FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);}
281  if (!open(FH_exc, ">$gexceptionsFile"))               {$eventType = 'FAIL'; $eventDesc = "MAIN::Cannot open GexceptionsFile: \[$gexceptionsFile]"; &log
     Event(\*FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);}
282
283  ## load %results fieldName keys from layout of file (now in %LAYOUT)
284
285  @fieldkeys    = undef;
286  @fieldkeys    = &getfieldkeys(\%LAYOUT);
287
288      #print "--debug--\n"; foreach(@fieldkeys) { print "**".$_."| \n"; }
289
290  $numberRequiredfields  = scalar(@fieldkeys);
291
292  if(!defined($LAYOUT{'orderProcessingDSN'})) { $LAYOUT{'orderProcessingDSN'} = $GDSN; }
293
294  print "** Loading daily order data source: $LAYOUT{'orderProcessingDSN'}...\n";
295  @DB   = &read_DSNtoAOH($LAYOUT{'orderProcessingDSN'},$LGDEBUG_FTN);
296
297  $totalRecords   = $#DB +1;
298  $gclientName    = $LAYOUT{'thisClientName'};
299
300  ## startup OK event
301
302  $eventType = '_X_INFO'; $eventDesc = "procOrder R.$grevision START OK - USER: $guser/$gcfPath CLIENT: $gclientName   opDSN ($totalRecords): $LAYOUT('orderP
     rocessingDSN') mdDSN: $LAYOUT('orderProcessingDSN') -> $LAYOUT('thisMailingTable')  ($totalRecords recs) BODYdefn: $gbodyDataLayoutFileName"; &logEvent(
     \*FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);
303
304  print "\n\n** status...\n\n";
305  print "USER:\t\t$guser \n";
306  print "CLIENT:\t\t$gclientName\n";
307  print "fPATH:\t\t$gcfPath\n\n";
308  print "TOTAL RECORDS\t\t -> \"$LAYOUT('orderProcessingDSN')\": $totalRecords\n";
309  unless($ADD_CHANGE_ON) { print "MAILING TABLE SOURCE \t -> \"$LAYOUT('thisMailingDSN') -> $LAYOUT('thisMailingTable')\"\n\n"; }
310  print "\n(Note: if this information is not correct, hit <CNTRL> - C to abort and correct <you have 5 seconds>)\n\n"; sleep 5;
311
312  print "** begin processing...\n\n";
313
314  ## flag duplicate e-mail addresses in order stream
315
316  %dupsMap = &checkdupsReturnMap(\@DB,$GDEBUG);
317
318  #################
319  #### main loop ###
320  #################
321
322  $excpCount           = 0;
323  $warnCount           = 0;
324  $outFilecount        = 0;
325  $exceptionsFileCount = 0;
326  $excReportFileCount  = 0;
327  $greportFilecount    = 0;
328  $FUPcount            = 0;
329  $reportLinecount     = 0;
330
331  for $mainCount (0 .. $#DB) {
332
```

```
     T:\Development\PRJPRJ_hbsp_extractOrder-WORKIN\G\procOrder.pl
     Printed at 19:48 on 06 Feb 1999

333  $thisRecordID                    = $maincount+1;
334  $eventRequiresFUP                = 1;
335  $RECORD_VALID                    = 1;
336  $thisBuyStatus                   = undef;
337  $thisShippingAddressBlock        = undef;
338  $thisBillingAddressBlock         = undef;
339  $thisMDBbillingAddressBlock      = undef;
340  $thisFullNumberedBody            = "\n";
341
342  ## get relevant fields, normalized, this record
343
344  $thisFromAddress                 = &normalizespacing(lc($DB[$maincount]{'FromAddress'}));
345  $thisFromAddress               =~ s/[\t\r]/$Gspace/g;
346
347  $thisFromName                    = &normalizespacing($DB[$maincount]{'FromName'});
348  $thisFromName                  =~ s/[\t\r]/$Gspace/g;
349
350  $thisSubject                     = &normalizespacing(uc($DB[$maincount]{'subject'}));
351  $thisSubject                   =~ s/[\t\r]/$Gspace/g;
352
353  $thisBody                        = uc($DB[$maincount]{'Body'});
354  $thisBody                      =~ s/[\t\r]/$Gspace/g;
355
356  ## split BODY into discrete lines
357
358  @BODYlines                       = undef;
359  @topLines                        = undef;
360  @bottomLines                     = undef;
361
362  push(@BODYlines, $1) while ($thisBody =~ s/^([^\012\015]*)(\012\015?|\015\012?)//);
363
364  ## skip lines that are the quoted BODY unless they're the order form -- example delimiters:
365  ## $BODYstartToken = 'HARVARD BUSINESS SCHOOL PUBLISHING';
366  ## $BODYendToken = 'FIRST NAME:';
367
368  $BODYstartToken                  = $LAYOUT{'startDelimiter'};
369  $BODYendToken                    = $LAYOUT{'endDelimiter'};
370  $BODYstartToken                =~ s/[\t\n\r]//g;
371  $BODYendToken                  =~ s/[\t\n\r]//g;
372  $startToken_ON                   = 0;
373  $endToken_ON                     = 0;
374  $thisBody                        = undef;
375  $thisBodyTop                     = undef;
376  $thisBodyBottom                  = undef;
377  $lastLineExcluded                = undef;
378
379  for($count = 0; $count <= $#BODYlines; $count++) {
380      $line                        = $BODYlines[$count];
381      $thisFullNumberedBody       .= ("[".$count."] ".$line."\n");
382
383
384      unless($startToken_ON || $line =~ /\^+/) {unless ($line !~ /\s+?/) {$thisBodyTop .= ($line.$Gpipe); push(@topLines,("[".$count."]  ".$line.'|')
385  );}
386
387      if(($line =~ /$BODYstartToken/) .. ($line =~ /$BODYendToken/)) {
388          if(($line =~ /$BODYstartToken/) && ($line =~ /$BODYstartToken/))   {$startToken_ON = 1;}
389          if(($line =~ /$BODYendToken/) && ($line =~ /$BODYendToken/))   {$endToken_ON = 1; $thisBody .= ($line.$Gnewline); unless($line =~ /\^+/
390  || $line !~ /\s+?/ || $line =~ /[\[\]]+/) {$thisBodyBottom .= ($line.$Gpipe)} }
391          next;
392      }
393
394      ## ignore bottom lines if part of the original -> indicators: ">" || "[" || "]"
395      if($startToken_ON && $endToken_ON) {unless($line =~ /\^+/ || $line !~ /\s+?/ || $line =~ /[\[\]]+/) {$thisBodyBottom .= ($line.$Gpipe); push(@bo
396  ttomLines,("[".$count."] ".$line.'|'));} }
397          $thisBody .= ($line.$Gnewline);
398  }
```

45

```perl
F:\Development\PRJ\PRJ_hbsp_extractOrder--WORKING\procOrder.pl
Printed at 19:48 on 06 Feb 1999

399  $thisBodyTop    =~ s/[\t\r]//g;
400  $thisBodyBottom =~ s/[\t\r]//g;
401
402  if($GDEBUG) { print "\n\n\n*** DEBUG_THISBODY=".$thisBody."\n\n\n"; }
403
404  ## log/print warning if cannot find FOOTER ID BLOCK: CUSTNO|LISTID|MAILINGID|DIALOGID || if bad match yields the same for CUSTNO|LISTID
405
406
407  $thisCustNO         = -1;
408  $thisListID         = -1;
409  $thisMailingID      = -1;
410  $thisDialogID       = -1;
411  $LISTCID_EXCEPT_ON  =  0;
412  $MAILINGID_EXCEPT_ON =  0;
413
414  if($thisBody !~ /\[\[([^\]]*?)\]\[([^\]]*?)\]\[([^\]]*?)\]?([^\]]*?)\]\/) {
415
416      $LISTCID_EXCEPT_ON = 1;
417
418      $eventType = "EX_ERR"; $excpCount++; $eventDesc = " \<".$thisRecordID."\>  UNMATCHED FOOTER ID BLOCK: CUSTNO|LISTID|MAILINGID|DIALOGID";
419      &logEvent(\*FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);
         $eventType = "INFO"; $excpCount++; $eventDesc = " \<".$thisRecordID."\> >>> ATTEMPTING SUBJECT LINE MAILING ID RECOVERY >>>"; &logEvent(
         \*FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);
420
421      ## test to see if there is a cf file synonym (**,++, etc) for the MAILING ID which can be recovered in the absence of the footer token
422
423      if($thisMailingID == -1} { $thisMailingID = &subjectSynonymLookup($thisSubject,\%LAYOUT); }
424      if($thisMailingID == -1} {
425
426          $RECORD_VALID=0;
427          $MAILINGID_EXCEPT_ON = 1;
428          $eventType = "EX_ERR"; $excpCount++; $eventDesc = " \<".$thisRecordID."\>  MAILING ID RECOVERY WAS NOT SUCCESSFUL"; &logEvent(\*F
         H_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);
429
430      }
431      if($thisMailingID != -1) {
432
433          $eventType = "INFO"; $excpCount++; $eventDesc = " \<".$thisRecordID."\>  <<< MAILING ID RECOVERY SUCCESSFUL USING: $thisSubjectMa
         ilingIDToken}"; &logEvent(\*FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);
434
435      }
436  }
437
438  else {
439
440      if($GDEBUG) { print "***DEBUG_ element1= $1 __ element2= $2 __ element3= $3 __ element4= $4\n"; sleep 1; }
441
442      $element1 = $1; $element2 = $2; $element3 = $3; $element4 = $4;
443
444
445      $thisCustNO    =~ {{$element1 =~ /\s+?/} ? $element1 : -1};
446      $thisListID    =~ {{$element2 =~ /\s+?/} ? $element2 : -1};
447      $thisMailingID =~ {{$element3 =~ /\s+?/} ? $element3 : -1};
448      $thisDialogID  =~ {{$element4 =~ /\s+?/} ? $element4 : -1};
449
450      if ($thisCustNO == -1 && $thisListID == -1) {
451
452          $LISTCID_EXCEPT_ON = 1;
453          $MAILINGID_EXCEPT_ON = 1;
454          $eventType = "EX_ERR"; $excpCount++; $eventDesc = " \<".$thisRecordID."\>  BAD MATCH:    CUSTNO, LISTID"; &logEvent(\*FH_log, $even
         tType, $eventDesc, $guser, $LGDEBUG_FTN);
         }
455
456      ## log/print INFO if cannot find mailingID
457
458      if($thisMailingID == -1) { $eventType = "INFO"; $eventDesc = " \<".$thisRecordID."\>  NO MAILING ID MATCH"; &logEvent(\*FH_log, $eventTyp
         e, $eventDesc, $guser, $LGDEBUG_FTN);}
459
460      ## log/print INFO if cannot find dialogID
461
```

```perl
462        if($thisDialogID == -1) { $eventDesc = " \<".$thisrecordID."\>    NO DIALOG ID MATCH"; &logEvent(\*FH_log, $eventType,
463    $eventDesc, $Guser, $LGDEBUG_FTN);}
464    }
465
466 ## check for /multi || quoted BUY TOKEN (eg: "YES")
467
468    $buyTokenPosition       = 'none';
469    $thisitemsselectedstack = undef;
470    @thisitemsselected      = undef;
471
472    unless($ADD_CHANGE_ON) {
473
474    ## check for buy tokens, eg: YES in SUBJECT, TOP, BOTTOM || letter items: ABC || A,B,C || A-C
475
476        $buyTokenPosition = &detectBuyTokens($MULTI_BUY_ON,$Gsubj_buyToken,$thissubject,$thismailingID,$thisbodyTop,$thisbodyBottom,\%LAYO
UT,\@topLines,\@bottomLines,$LGDEBUG);
477
478    ## check for no-match case and except
479
480    if($buyTokenPosition eq 'none') { $eventRequiresFUP = '.noTok'; $eventType = 'ERR_FUP'; $warnCount++; $eventDesc =
"\<".$thisrecordID."\> SUBJECT, TOP, BOTTOM -> NO BUY TOKEN MATCH ($Gsubj_buyToken)"; &logevent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN)
;  }
481    ## if match and MULTI_BUY_ON, prepare order items listing
482
483    if($buyTokenPosition ne 'none' && $MULTI_BUY_ON) {
484
485        $thisitemsselectedstack = &extractDedupItemsselectedstack(@thisitemsselected);
486
487    }
488
489    }
490 ###############
491 ## INNER LOOP ##
492 ###############
493
494
495    %results          = {};
496    $countFields      = 0;
497    $mulTcriticalFields = 0;
498    $ADDRESS_ADEQUATE = 1;
499
500 ## process body fields by key; keys == standard field names; values == this BODY's field names mapped thereon
501    foreach (@fieldkeys) {
502
503    $key = $_;
504
505    ## print "--DEBUG-- key = |$key| \n";
506
507
508    $thislayoutfieldname = uc($LAYOUT{$key});
509    $thislayoutfieldname =~ s/\s+$//;
510
511 ##### #########################
512 ##### MAIN MATCHING BLOCK #######
513 ##### #########################
514
515    $MATCH_DATA_ON = 0;
516    $RULE = -1;
517
518    $expectedMatch   = $thisLayoutfieldname."\:.*?){[(A\][)+?\]\":";
519    $noBracketsMatch = $thisLayoutfieldname."\:.*?){[A\(0\]2\(0\]5\]+?)";
520
521    CASE: {
522
523    TRY_EXPECTED_MATCH:    if($thisBody =~ /$expectedMatch/)   { $MATCH_DATA_ON = 1; $matchElement = $1; $RULE = 1; last
523    TRY_NO_BRACKETS_MATCH: if($thisBody =~ /$noBracketsMatch/) { $MATCH_DATA_ON = 1; $matchElement = $1; $RULE = 2; last
524    CASE: }
524    CASE: }
```

```perl
525         }
526
527 ## A MATCH WAS FOUND, THIS KEY ... push into results set
528
529     if($MATCH_DATA_ON) {
530
531             $countFields++;
532             $matchElement  =~ s/[A\w\s\-\.\/\#\!\@]/$Gspace/g;
533             $results{$key} = &normalizespacing($matchElement);
534
535 ## print "--DEBUG-- result = |$results{$key}| \n";
536
537 ## build shipping and billing address blocks
538
539         CASE: {
540
541             if($key =~ /u[a-z]_/) {
542
543                 ## construct SHIPPING ADDRESSBLOCK for report
544
545                 if ($key =~ /[hijklmq]_/) { $thisShippingAddressBlock .= (length($results{$key}) > 0)?($results{$key}.$Gpipe):''; }
546                 if ($key =~ /[efgnop]_/) { $thisShippingAddressBlock .= (length($results{$key}) > 0)?($results{$key}.$Gspace):''; }
547             }
548             last CASE;
549         }
550
551             if($key =~ /[^u]_/) {
552
553                 ## construct BILLING ADDRESSBLOCK for report
554
555                 if ($key =~ /[hijklmq]_/) { $thisBillingAddressBlock .= (length($results{$key}) > 0)?($results{$key}.$Gpipe):''; }
556                 if ($key =~ /[efgnop]_/) { $thisBillingAddressBlock .= (length($results{$key}) > 0)?($results{$key}.$Gspace):''; }
557             }
558             last CASE;
559         }
560
561         }
562
563     ## check for null critical fields: any absence here will throw exception, unless CUSTNO
564
565 ## print "--debug-- |$LAYOUT{'criticalFields'}| \n":sleep 1;
566
567         if ($key =~ /($LAYOUT{'criticalFields'})_/ && $results{$key} !~ /\s+?/ && $thisCustNo < 1) {
568
569             $RECORD_VALID=0; $ADDRESS_ADEQUATE = 0;
570             $eventRequiresFUP .= (($eventRequiresFUP =~ /badAd/)?'':'badAd'); $FUPcount++; $eventType = 'EX_FUP'; $excpCount++; $even
571             CRITICAL STANDARD FIELD_VALUE IS NULL: \[$thisLayoutFieldname\]"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $L
GDEBUG_FTN);
572 tDesc = " \<".$thisRecordID."\>
573
574         }
575
576
577 ## ALL MATCHES FAILED, THIS KEY
578
579         if(!$MATCH_DATA_ON) {
580
581             $results{$key} = undef;
582             $eventType = 'EX_ERR'; $excpCount++; $eventDesc = " \<".$thisRecordID."\>    STANDARD FIELD_NAME=FIELD_VALUE PAIR UNPARSABLE: \[$th
isLayoutFieldname\]"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
583
584         # check to see is unparsable field is a critical field
585         # critical fields -> from CF file: any absence here will throw exception
586
```

48

```
File Development\PROJ...jhbsp extractOrder...WORKINGprodOrder.pl
Printed at 19:48 on 05 Feb 1999

587     if ($key =~ /[$layout{'criticalFields'}]\/>) {
588         $RECORD_VALID=0; $ADDRESS_ADEQUATE = 0;
589         $eventRequiresFUP .= (($eventRequiresFUP =~ /badAd/)?'':'.badAd'); $FUPcount++; $eventType = 'EX_FUP'; $excpCount++; $even
590     tDesc = " \<".$thisRecordID."\> UNPARSABLE STANDARD FIELD_NAME=FIELD_VALUE PAIR IS CRITICAL: \[$thislayoutfieldname\]"; &logEvent(\*FH_log, $eventType,
        $eventDesc, $Guser, $LGDEBUG_FTN);
591     }
592
593     }
594
595     #####################
596     ### END INNER LOOP ###
597     #####################
598
599     #####################
600     #####################
601     ### SUMMARY RULES ###
602     #####################
603     ## throw EX_LISTNO/CUSTNO error exception if address is not OK && !phone, depending on CID_LID_REQUIRED from cf file
604
605     if(!$layout{'CID_LID_REQUIRED'}) {
606
607         if($LISTCID_EXCEPT_ON) {
608
609             $RECORD_VALID = 0;
610             $eventRequiresFUP .= (($eventRequiresFUP =~ /chkCLID/)?'':'.chkCLID'); $FUPcount++; $eventType = 'EX_FUP'; $excpCount++; $eventDes
611     c = " \<".$thisRecordID."\> EITHER CID || LID ABSENT WHEN CF CRITICAL"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
612
613         }
614
615     }
616
617     if(!$layout{'CID_LID_REQUIRED'}) {
618
619         if($LISTCID_EXCEPT_ON && !($ADDRESS_ADEQUATE) && $results{'r\_phone'} !~ /\S+?/)) { $RECORD_VALID = 0; }
620
621     }
622
623     ## throw EX_MAILINGID exception if cf file says it's required
624
625     if(!$layout{'MAILINGID_REQUIRED'}) {
626
627         if($MAILINGID_EXCEPT_ON) {
628
629             $RECORD_VALID = 0;
630             $eventRequiresFUP .= (($eventRequiresFUP =~ /chkMID/)?'':'.chkMID'); $FUPcount++; $eventType = 'EX_FUP'; $excpCount++; $eventDesc
631     = " \<".$thisRecordID."\> MAILING ID ABSENT WHEN CF CRITICAL"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
632
633         }
634
635     }
636
637     ## throw INFO if less than number required
638
639     if(($countFields < $numberRequiredFields) && $thisCustNo < 1) {
640         $eventType = 'INFO'; $eventDesc = " \<".$thisRecordID."\>    (OVERALL) NOT ALL STANDARD FIELDS FOUND: $countFields ($numberRequiredFields)"
641     ; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
642
643     ## tag duplicates for FUP
644
645     if($dupsMap{$mainCount} =~ /\f*\f.+\f*\f/) { $eventRequiresFUP .= (($eventRequiresFUP =~ /dupAd/)?'':'.dupAd'); $FUPcount++; }
646
647     ## catch thisBodyTop and thisBodyBottom blank yet address still extracted adequately (eg happens when cut & paste responses between the body delimiters)
648
649     if($RECORD_VALID && $thisBodyTop && $thisBodyTop !~ /\S+?/ && $thisBodyBottom !~ /\S+?/) {
650
```

49

```
651        $RECORD_VALID = 0;
-652       $eventType = 'EX_ERR'; $eventRequiresFUP .= '.chkBody'; $FUPcount++; $excpcount++; $eventDesc = " \<".$thisRecordID."\> CUSTOMER RESPONSE (TOP.
           Bottom) NOT FOUND WHEN EXPECTED"; &logEvent(\*FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);
653      }
654      ############################################################
655      ############################################################
656      #### RECORD INVALID: Attempt recovery from mailing database ####
657      ############################################################
658      ############################################################
659      $RECORD_RECOVERED = 0;
660
661      if(!$RECORD_VALID && !$ADD_CHANGE_ON) {
662        $eventType = 'INFO'; $eventDesc = " \<".$thisRecordID."\> >>> BAD RECORD: ATTEMPTING RECOVERY-LOOKUP USING (\"$thisFromAddress\", $LAYOUT{'thisMa
663        ilingDSN'} -> $LAYOUT{'thisMailingTable'}"; &logEvent(\*FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);
664
665        %recoveredresults = undef;
666        %recoveredresults = &querySELECT_DB($LAYOUT, 'EMAIL', $thisFromAddress, $LGDEBUG_FTN);
667
668        ## foreach(keys(%recoveredresults)) { print "=debug== key = $_ ___ value = $recoveredresults{$_} \n"; }
669
670        CASE: {
671
672          if($recoveredresults{'_exit_'} < 1) {
673
674            $RECORD_VALID = 0;
675            $eventType = 'EX_ERR'; $eventRequiresFUP .= /badMDB/)?'':'.badMDB') ; $FUPcount++; $eventDesc, $eventDesc, $guser, $LGDEBUG_F
676            = " \<".$thisRecordID."\> MAILING TABLE RECOVERY-LOOKUP FAILED ON \"$thisFromAddress\""; &logEvent(\*FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);
677            TN);
678          last CASE;
679          }
680
681          if($recoveredresults{'_exit_'} == 1) {
682
683            $RECORD_RECOVERED = 1;
684            $RECORD_VALID = 1;
685            $eventType = 'WRN_FUP'; $eventRequiresFUP .= /goodMDB/)?'':'.goodMDB'); $FUPcount++; $excpcount++; $eventDe
               sc = " \<".$thisRecordID."\> <<< MAILING TABLE RECOVERY-LOOKUP SUCCESSFUL USING \"$thisFromAddress\""; &logEvent(\*FH_log, $eventType, $eventDesc, $guse
               r, $LGDEBUG_FTN);
686            ## construct $thisMDBbillingAddress
687
688            foreach $key (sort(keys($MDBLAYOUT))) {
689
690              $value = $MDBLAYOUT{$key};
691
692              ## print "**DEBUG key = $key ___ value = $value \n";
693
694              if ($key =~ /[hijklmq]\_/) { $thisMDBbillingAddressBlock .= (length($recoveredresults{$value}) > 0)?($recoveredresults{$va
695              if ($key =~ /[efgnop]\_/) { $thisMDBbillingAddressBlock .= (length($recoveredresults{$value}) > 0)?($recoveredresults{$val
696            ue}).$gpipe):'';  }
               ue}).$gspace):'';  }
697            }
698
699            ## restore mailingID and CID, if blank
700
701          ## print "**DEBUG mailingID = $recoveredresults{'ED_MAILINGID'} ___ custno = $recoveredresults{'CID'} ___ current custno = |$thisCustNo|\n";
702            if(($thisMailingID !~ /\S+?/ || $thisMailingID == -1)    { $thisMailingID   = $recoveredresults{'ED_MAILINGID'}; }
703            if(($thisCustNo !~ /\S+?/ || $thisCustNo == -1)          { $thisCustNo      = $recoveredresults{'CID'}; }
704
705          last CASE;
706          }
707        }
708
709      }
710
711
```

50

```
712   }
713
714   #### add an empty flag where SHIPPING ADDRESS is returned blank
715
716   if($thisshippingAddressBlock !~ /\s+?/){ $thisshippingAddressBlock = '{ SAME }'; }
717   if($thisMDBbillingAddressBlock !~ /\s+?/){ $thisMDBbillingAddressBlock = '{ N/A }'; }
718
719   #### tag address block exceptions with diagnostic ques
720
721   if(!$RECORD_VALID) {
722
723       CASE: {
724
725           if($thisBillingAddressBlock !~ /\s+?/ && ($thisBodyTop =~ /\s+?/ || $thisBodyBottom =~ /\s+?/))
                  { $thisBillingAddressBlock = 'NO_
                  VALID_ADD/INSPECT='; last CASE;}                              { $thisBillingAddressBlock = '{ BO
726           if($thisBillingAddressBlock !~ /\s+?/ && $thisBodyTop !~ /\s+?/ && $thisBodyBottom !~ /\s+?/)
                  DY_EMPTY/INSPECT'; last CASE;}
727           if($thisBillingAddressBlock =~ /\s+?/ && !$ADDRESS_ADEQUATE)     { $thisBillingAddressBlock = 'CRIT
                  _FLD_NUL/INSPECT ->|'.$thisBillingAddressBlock.'|'; last CASE; }
728
729               $thisBillingAddressBlock = 'UNKNOWN_ERR/INSPECT=|'.$thisBillingAddressBlock; last CASE;
730
731       }
732
733       $eventRequiresFUP = ($eventRequiresFUP !~ /\s+?/) ? '.inspect' : $eventRequiresFUP;
734   }
735
736   ###############################################
737   ## write valid record data to outfiles ##
738   ###############################################
739
740   $ED_ORDER_VALUE = 1;
741   $EXCEPTION_TYPE = 'OK';
742   $UPDATE_SUCCESSFUL = 0;
743   $THIS_KEY_FIELD = undef;
744   $THIS_KEY_FIELD_VALUE = undef;
745
746   if($RECORD_VALID) {
747
748       unless($ADD_CHANGE_ON) {
749
750           CASE: {
751
752               if(&queryUPDATE_DB(\%LAYOUT,$THIS_KEY_FIELD_VALUE = $thisFromAddress,$THIS_KEY_FIELD = "EMAIL","ED_ORDER","ED_ORDER",$ED_ORDER_VALUE,
                  "ED_MAILACTION",-1,$RECORD_RECOVERED,$LGDEBUG_FTN)) { $UPDATE_SUCCESSFUL=1; last CASE; };
                  if(&queryUPDATE_DB(\%LAYOUT,$THIS_KEY_FIELD_VALUE = $thisCustno,$THIS_KEY_FIELD = "CID","ED_ORDER",$ED_ORDER_VALUE,"ED_MAI
753           LACTION",-1,$RECORD_RECOVERED,$LGDEBUG_FTN)) { $UPDATE_SUCCESSFUL=1; last CASE; };
                  if(&queryUPDATE_DB(\%LAYOUT,$THIS_KEY_FIELD_VALUE = $thisListID,$THIS_KEY_FIELD = "ED_LID","ED_ORDER",$ED_ORDER_VALUE,"ED_
754           MAILACTION",-1,$RECORD_RECOVERED,$LGDEBUG_FTN))     { $UPDATE_SUCCESSFUL=1; last CASE; };
755
756           }
757
758       }
759
760       if(!$UPDATE_SUCCESSFUL) {
761
762           $RECORD_VALID = 0;
763           $eventType = 'EX_ERR'; $eventRequiresFUP .= (($eventRequiresFUP =~ /badupMDB/)?'':'.badupMDB') ; $FUPcount++; $excpCount++; $event
              Desc = "\<".$thisRecordID."\> (MANUAL UPDATE REQUIRED) MAILING TABLE UPDATE FAILED ON ALL KEYS - LAST TRY: \"$THIS_KEY_FIELD -> $THIS_KEY_FIELD_VALUE\"
              "; &logevent(\*FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);
764
765       }
766
767   }
768
769   if(!$RECORD_VALUE && !$UPDATE_SUCCESSFUL) {
770
771       $eventType = 'EX_ERR'; $eventRequiresFUP .= (($eventRequiresFUP =~ /noupMDB/)?'':'.noupMDB') ; $FUPcount++; $guser, $LGDEBUG_FTN);
       $eventRequiresFUP = ($eventRequiresFUP =~ /noupMDB/)?'':'.noupMDB') ; $FUPcount++; $excpCount++; $eventDesc = " \
       <".$thisRecordID."\> INVALID RECORD: MAILING TABLE NOT UPDATED"; &logevent(\*FH_log, $eventType, $eventDesc, $eventDesc = " \
```

51

```
772  }
773
774
775  #### Buy tokens not found and not ADDRESS_CHANGE; move to EXC report
776      unless($ADDRESS_CHANGE) { if($buyTokenPosition eq 'none') { $RECORD_VALID = 0; $ED_ORDER_VALUE = 2; } }
777
778      if($RECORD_VALID) {
779
780          $reportlineCount++;
781
782          ## update MAILING TABLE
783
784
785          ## write out files
786
787          &writeRecord(\*FH_out,$Gfieldelimiter,($thiskey=&getNextDBkey_return($GoutFileDBkeysFilename,$LGDEBUG_FTN)),\%GoutFilerecord,'EXTENDED',
788  $LGDEBUG_FTN);
789          &writeRecord(\*FH_report,$Gfieldelimiter,($thiskey=&getNextdBkey_return($GreportFileDBkeysFilename,$LGDEBUG_FTN)),\%GreportFilerecord,'NO
790  RMAL', $LGDEBUG_FTN);
791          ## include ALL BODYs in the EXC_BODY file for potential review
792
793          if($eventRequiresFUP =~ /\s+?/) { $EXCEPTION_TYPE = 'SOFT-EXC'; }
794
795              &writeRecord(\*FH_excBODYReport,$Gfieldelimiter,$thisBODYlogkey=&getNextDBkey_return($GexceptionsBODYDBkeysFilename,$LGDEBUG_FTN)
796  ,\%GexceptionsBODYFilerecord,'BODY', $LGDEBUG_FTN);
797          $outFilecount++;$reportFilecount++;
798      }
799
800  ###############################################
801  ## write exception record data to outfiles ##
802
803  ###############################################
804
805      if(!$RECORD_VALID) {
806
807          $EXCEPTION_TYPE = 'HARD-EXC';
808
809          &writeRecord(\*FH_exc,$Gfieldelimiter,$thiskey=&getNextDBkey_return($GexceptionsDBkeysFilename,$LGDEBUG_FTN),\%GoutFilerecord,'EXTENDED',
810  RMAL',$LGDEBUG_FTN);
811          &writeRecord(\*FH_excReport,$Gfieldelimiter,$thiskey=&getNextDBkey_return($GexcReportDBkeysFilename,$LGDEBUG_FTN),\%GreportFilerecord,'NO
     RMAL',$LGDEBUG_FTN);
         &writeRecord(\*FH_excBODYReport,$Gfieldelimiter,$thisBODYlogkey=&getNextDBkey_return($GexceptionsBODYDBkeysFilename,$LGDEBUG_FTN),\%Gexce
     ptionsBODYFilerecord,'BODY', $LGDEBUG_FTN);
812          $exceptionsFilecount++; $excReportFilecount++;
813
814      }
815
816      print "\n";
817  }
818
819  ###########################
820  #### end main loop ####
821  ###########################
822
823
824  ## print STDOUT exit state
825  $dateNow = &getDateStamp('WIN',$LGDEBUG_FTN);
826
827  print "\n** [procOrder version $Grevision] Done! $totalRecords records processed by $Guser/$GcfPath for client: $Gclientname **\n\n";
828  print "SUMMARY REPORT for:" .$dateNow. "\n\n";
829  print "FILES WRITTEN \n";
830  printf "\n%10d ",$reportFilecount;
831  print " records written to REPORT FILE: $Greportfile";
832  printf "\n%10d ",$excReportFilecount;
833  print " records written to EXC_REPORT FILE: $excReportfile";
834  printf "\n%10d ",$outFilecount;
```

52

```
     F:\Development\PRJ\PRJ_hbsp_extractOrder-WORKING\proc0rder.pl
     Printed at 10:48 on 06 Feb 1999

835    print " records written to OUTPUT FILE (all fields): $coutFile";
836    printf "\n%10d",$exceptionsFilecount;
837    print " records written to EXCEPTIONS FILE (all fields): $GexceptionsFile";
838    print "\n\nLOGGING\n\n";
839    print "All rule failure events for all time logged in detail to: $GactivityLogFileName\n";
840    printf "\n%10d",$totalRecords;
841    print " BODYs for this session are keyed, categorized in: $GexcBODYReportFile, for review in an editor\n";
842
843    $eventType = '_x_INFO'; $eventDesc = "procOrder R.$Grevision EXIT OK - USER: $Guser/$GcfPath CLIENT: $GclientName     opDSN ($totalRecords): $LAYOUT('orderPr
       ocessingDSN') mdbDSN: $LAYOUT('orderProcessingDSN') -> $LAYOUT('thisMailingTable') ($totalRecords recs) BODYdefn: $GbodyDataLayoutFileName"; &logEvent(\*
       FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);

844    close(FH_out); close(FH_exc); close(FH_log); close(FH_report); close(FH_excReport);close(FH_excBODYReport);
845    ######################################################
846    ##################### SUBS ###########################
847    exit;
848    ######################################################
849    ######################################################
850    ##################### SUBS ###########################
851    ######################################################
852    #----------------------------------------------------
853    #
854    #
855    #
856    #----------------------------------------------------
857    sub alphaOrdered {
858
859    my(@singleElementArray) = @_;
860
861    my $alpha    = 0;
862    my $i        = 0;
863
864    my $base = $singleElementArray[0];
865
866    for($i = 1; $i <= $#singleElementArray; $i++) {
867
868        $ordBase = ord($base); $ordArr = ord($singleElementArray[$i]);
869
870    ##     print "==debug=$i= base = |$base| ordBase= $ordBase ____ singleElementArray = $singleElementArray[$i] ordArr = $ordArr \n";
871
872        if($ordBase > $ordArr) { return $alpha; }
873        $base = $singleElementArray[$i];
874
875    }
876    $alpha = 1;
877
878    return $alpha;
879
880    }
881    #----------------------------------------------------
882    #
883    #
884    #----------------------------------------------------
885    sub canonicalize {
886
887    my($inRecord) = @_;
888    my($elementsIndex, $fieldsIndex, $rec, $buf, $field) = undef;
889    my(@field) = undef;
890    my $space = '_';
891    my $dashEscape = $space.'_d_esc_'.$space;
892
894    $inRecord = uc($inRecord);
895
896    my @allFields = undef;
897
898    @allFields = split(/$Gfieldelimiter/,$inRecord);
899    my $fieldsIndex = undef;
900
901    for($fieldsIndex=0; $fieldsIndex <= $#allFields; $fieldsIndex++) {
```

53

```perl
E:\Development\PRJPRJ hbsp extractOrder-WORKINGhbr\Order.pl
Printed at 19:48 on 06 Feb 1999

902    if ($fieldsIndex != $emailFieldpos && $fieldsIndex != $fromAddressFieldpos) {
903
904      if( $fieldsIndex == $companyFieldpos || $fieldsIndex == $lastNameFieldpos || $fieldsIndex == $firstNameFieldpos || $fieldsIndex == $titl
905      efieldpos || $fieldsIndex == $countryFieldpos || $fieldsIndex == $departmentFieldpos || $fieldsIndex == $cityFieldpos) { $allFields[$fieldsIndex] =~ s/\
         -/$dashEscape/g; }

906      $allFields[$fieldsIndex] =~ s/\s+$//;                    # trailing space
907      $allFields[$fieldsIndex] =~ s/^\s+//;                    # leading space
908      $allFields[$fieldsIndex] =~ s/[\.,;\\\/()]/$space/g;     # extraneous chars
909      $allFields[$fieldsIndex] =~ s/\s{2,}/$space/g;           # more than one separating space
910
911    }
912    elsif ($fieldsIndex == $emailFieldpos || $fieldsIndex != $fromAddressFieldpos) { $allFields[$fieldsIndex] =~ s/$space//g; }
913
914    @field = split($space,$allFields[$fieldsIndex]);
915
916    $field = undef;
917
918    for($elementsIndex = 0; $elementsIndex <= $#field; $elementsIndex++) {
919
920      CONVERT_TITLE_CASE: {
921
922        $leaveupper = 1;
923
924        if($fieldsIndex == $postalCodeFieldpos) { last CONVERT_TITLE_CASE; }
925        if($headerFields[$fieldsIndex] !~ /\w*IX\w*$/ && $field[$elementsIndex] =~ /^[AAEIOU\.][AAEIOU\.]?$/) { last CONVERT_TIT
926        LE_CASE; }
927
928        if($field[$elementsIndex] =~ /^[A-Z]\.[A-Z]\.[A-Z]\.[A-Z]?\.?$/) { last CONVERT_TITLE_CASE; }
929
930        unless($field[$elementsIndex] =~ /(AND)|(OR)|(UND)|(AT)|(CO)|(LTD)|(INC)|(OF)|(THE)|(RD)/) {
931
         if($field[$elementsIndex] == $firstNameFieldpos && ($field[$elementsIndex] =~ /^[A-Z][A-Z]s/ || $field[$elementsIndex] =~
         /^[A-Z]\.[A-Z]?$/)) { last CONVERT_TITLE_CASE; }
932        if($fieldsIndex == $ixFieldsIndex == $companyFieldpos && ($field[$elementsIndex] =~ /^[A-Z]\.?[A-Z]\.?[A-Z]\.?[A-
         z]s/ || $field[$elementsIndex] =~ /^[A]s]\.?[A-z]\.?s/) { last CONVERT_TITLE_CASE; }
933        tsIndex] !~ /[MD][RS]/)   if($fieldsIndex == $titleFieldpos && $field[$elementsIndex] =~ /^[ARHMGIVUCS][EMSQ&TRAFIOPV][PADO\.]*s/ && $field[$elemen
934        tsIndex] !~ /[MD][RS]/)   { last CONVERT_TITLE_CASE; }
         - /^[A-Z]\.[A-Z]?$/) { if($field[$elementsIndex] =~ /^[A-Z]\.[A-Z]\.?[A-Z]\.?$/ || $field[$elementsIndex] =
935      }
936
937      unless($fieldsIndex == $stateFieldpos && length($field[$elementsIndex])>5) { if($fieldsIndex == $stateFieldpos || $fieldsIndex ==
938      $postalCodeFieldpos)   { last CONVERT_TITLE_CASE; } }
939
940        $leaveupper = 0;
941
942        $buf = lc($field[$elementsIndex]);
943
944        if($fieldsIndex != $emailFieldpos && $fieldsIndex != $fromAddressFieldpos)   { $buf = ucfirst($buf); last CONVERT_TITLE_CASE; }
945
946      }
947      if($leaveupper) { $buf = $field[$elementsIndex]; }
948    }
949
950      $field .= ($buf.$space);
951    }
952
953    $rec .= ($field.$cFieldDelimiter);
954  }
955  chop($rec);
956
957  $rec =~ s/$dashEscape/\-/g;
958
959  return $rec;
960 }
961 }
962 }
```

54

F:\Development\PR\PRJ_hbsp_extractOrder—WORKIN\S\procOrder.pl
Printed at 10:48 on 06 Feb 1999

```perl
963   #
964   #-----
965   #
966   #
967   sub checkDupsReturnMap {
968
969   my($ptrAOHDB,$GDEBUG)   = @_;
970   my(@DB)                 = @$ptrAOHDB;
971
972   ## prepare map of duplicates on FromAddress
973
974   my %dupsMap
975   my $j
976
977   for $j (0 .. $#DB) {
978
979        $dupsMap{$j} = $DB[$j]{'FromAddress'}
980
981   }
982
983   my @mirrorDupsMap       = undef;
984   @mirrorDupsMap          = values(%dupsMap);
985   my $key                 = undef;
986   my $value               = undef;
987   my @arr                 = undef;
988   my $number              = undef;
989
990   foreach $key (keys %dupsMap) {
991
992        $value = $dupsMap{$key};
993        next if($value !~ /\S+?/);
994
995        @arr    = undef;
996        @arr    = grep /$value/, @mirrorDupsMap;
997        $number = scalar(@arr);
998
999        if($number > 1) {
1000
1001  g,  $eventType, $eventDesc, $guser, $GDEBUG_FTN);
           $eventType, $eventDesc = 'INFO', $eventDesc = "DUPLICATE FromAddress ITEM DETECTED ON IMPORT & FLAGGED: ($number|$key|$value)"; &logEvent(\*FH_lo
           $dupsMap{$key} = ('###'.$number.'### '.$dupsMap{$key});
1002
1003       ## print $dupsMap{$key}."\n";
1004
1005       }
1006       ## print "*** key=".$key." *** value=".$dupsMap{$key}."__occurrences=".$number."\n";
1007
1008   }
1009
1010   return %dupsMap
1011   }
1012   #
1013   #-----
1014   #
1015   #
1016   sub checkMultiBuy_Extract {
1017
1018   my($thisLine,$thisMailingID,$pos,$ptrLAYOUT,$GDEBUG) = @_;
1019   my($exit) = 0;
1020
1021   my(%LAYOUT)                  = %$ptrLAYOUT;
1022   my(@rangeStack)             = undef;
1023   my(@singleStack)            = undef;
1024   my($countRangeMatches)      = 0;
1025   my($repeat)                 = 1;
1026   my($match)                  = undef;
1027   my($replacementsChar)       = '%';
1028   my($preprocRemoveChar)      = '^';
1029   my($elementSeparator)       = '';
```

```
      F:\ Development\PPR\VPR\_hbsp\extractOrder--WORKING\ExtractOrder.pl
      Printed at 19:41 on 06 Feb 1999

1031
1032  ## PREPARE THE LINE FOR MATCHING
1033
1034  $thisline                 = uc($thisline);
1035
1036  ## RUN RULESET FROM RULES CF FILE
1037
1038  foreach(@GpreprocRules) { eval; }
1039
1040  my $thisPreExtractionLine         = $thisline;
1041  my @testsinglestack               = undef;
1042  my $maxASCII                      = 0;
1043
1044  ## EXTRACTION LOOP
1045
1046  while($repeat) {
1047
1048  my $RULE = 0;
1049  my $match = undef;
1050
1051  ##print "\n\n--debug-- maxASCII = $maxASCII -- thisline IS CURRENTLY |$thisline| \n\n";
1052
1053       MATCH_CASE: {
1054
1055           SINGLE_CASE: {
1056
1057  ##              if($thisline =~ /^([A-Z])$/)              { $match = $1; $RULE = -1;  last MATCH_CASE; }
1058                 if($thisline =~ /^([A-Z])[{(A-Z\\&\@\-]{1}/)   { $match = $1; $RULE = -2;  last MATCH_CASE; }
1059                 if($thisline =~ /[(A-Z)\A\&\@\-]{1}[A-Z\\&\@\-]{1}[A-Z\\&\@\-]*/)  { $match = $1; $RULE = -3;  last MATCH_CASE; }
1060
1061           }
1062
1063           RANGE_CASE: {
1064
1065                 if($thisline =~ /^([A-Z]\-[A-Z])[^A-Z\\&\@]{1}/)           { $match = $1; $RULE = 1;  last MATCH_CASE; }
1066                 if($thisline =~ /[(A-Z\\&)\@]{1}[(A-Z)\-[A-Z]}[(A-Z\\&\@]{1}/)  { $match = $1; $RULE = 2;  last MATCH_CASE; }
1067  ##             if($thisline =~ /[(A-Z]{1}[A-Z\\&]{1}[A-Z\\&\@]{1}/)     { $match = $1; $RULE = 3;  last MATCH_CASE; }
1068  ##             if($thisline =~ /[(A-Z]{1}[A-Z]*/)                        { $match = $1; $RULE = 4;  last MATCH_CASE; }
1069
1070           }
1071
1072       }
1073  $repeat = 0;
1074  }
1075  $thisline =~ s/$match/$replacementeschar/g;
1076
1077  $thischeckedMailingID = ($thisMailingID=0) ? $thisMailingID : -1;
1078
1079  ##print "\n-- debug - ELEMENT CONSIDERED with break = $RULE |$match| \n";
1080  if($GDEBUG) { print "--debug-- BEGIN RULES -- RANGE = |$LAYOUT{$thischeckedMailingID}| from $thischeckedMailingID \n"; }
1081
1082  ## BEGIN RULES TO SEE WHAT WE GOT ##
1083
1084  $PUSH = 1;
1085
1086  if($RULE > 0) {
1087
1088       my(@arr) = undef;
1089       my $tempmatch = $match;
1090       my $tempmatch =~ s/\-//g;
1091       @arr = split(//,$tempmatch);
1092
1093       if(!&alphaOrdered(@arr)) {
1094
1095  ##  print "-- debug -- REJECT - not alpha match |$match| in line |$thisPreExtractionLine| \n";
1096           $PUSH = 0;
1097
1098       }
1099  }
```

56

```
     F:\Development\PRJ\PRJ_hb\p_extractOrder-WORKING\procOrder.pl
     Printed at 19:48 on 06 Feb 1999

1100     }
1101
1102   #if($RULE > 0 && ord($arr[0]) <= $maxASCII) {
1103
1104 #        print "-- debug -- range case |$match| out of alpha order with current stack!\n";
1105 #        if(ord($match)>$maxASCII) {$maxASCII = ord($match); }
1106 #        $PUSH = 0;
1107
1108   #}
1109
1110   if(defined($match) && $match !~ /[$LAYOUT{$thisCheckedMailingID}]/) {
1111
1112 ##     print "-- debug-- REJECT - not in RANGE = |$LAYOUT{$thisCheckedMailingID}| match |$match| in line |$thisPreExtractionLine|   \n" ;
1113        if(ord($match)>$maxASCII) {$maxASCII = ord($match); }
1114        $PUSH = 0;
1115
1116     }
1117
1118   if($PUSH && $RULE > 0 && $match =~ /\S+?/) {
1119
1120 ##        push(@rangestack, ('<'.$RULE.'>'.$match));
1121           $match =~ s/\~/thru /g;
1122           push(@rangestack, $match);
1123           $exit=1;
1124
1125     }
1126
1127   if($PUSH && $RULE < 0 && $match =~ /\S+?/) {
1128           push(@testsinglestack,$match);
1129
1130        unless(ord($match) <= $maxASCII) {
1131
1132 ##           push(@singlestack, ('<'.$RULE.'>'.$match));
1133              push(@singlestack,$match);
1134              $exit=1;
1135
1136              if(ord($match)>$maxASCII) {$maxASCII = ord($match); }
1137
1138        }
1139     }
1140
1141        if($GDEBUG) { if(!defined($match)) {print "--debug-- match not defined! in line= |$thisPreExtractionLine|\n"; }
1142
1143     }
1144   ## consolidate extractions, pre report
1145
1146   foreach(@rangestack)    { unless($_ !~ /\S+?/) { push(@thisItemsselected,$_); }}
1147   foreach(@singlestack)   { unless($_ !~ /\S+?/) { push(@thisItemsselected,$_); }}
1148
1149   ## SPECIAL POST EXTRACTION CASE RULES
1150
1151        my $FIND_BUY_ALL = 0;
1152
1153   foreach(@GBuyAllIndicators) { s/[\n\r]//g; next if($_ !~ /\S+?/); if($thisPreExtractionLine =~ /$_/) { $FIND_BUY_ALL = 1; } }
1154
1155     if ($FIND_BUY_ALL) {
1156
1157        if($exit) {
1158
1159           my $t = '.vfytok';
1160           $eventType = 'WRN_FUP'; ($eventRequiresFUP !~ /$t/) ? $eventRequiresFUP .= $t ; $eventRequiresFUP ; $FUPCount++; $excpCount++; $e
     ventDesc = " \<".$thisRecordID."\> checkMultiBuy_EXTRACT :: AMBIGUOUS ALL-ITEM SELECTION DETECTED IN EXTRACTION LOCATION <$pos>"; &logEvent(\*FH_log, $e
     ventType, $eventDesc, $Guser, $LGDEBUG_FTN);
1161
1162           push(@thisItemsselected," ALL ($thisMailingID) "); $exit=1;
1163
1164        }
1165     }
1166
```

57

```
1167    ## DETECT DIFFERENCE IN EXTRACTION STACK AND MAX EXTRACTION STACK
1168
1169    if ($exit && (scalar(@singlestack) != scalar(@testsinglestack)) ) {
1170
1171            my $t = '.vfyTok';
1172            $eventType = 'WRN_FUP'; ($eventRequiresFUP !~ /$t/) ? $eventRequiresFUP .= $t: $eventRequiresFUP ; $FUPcount++; $excpCount++; $eventDesc
        = "\<".$thisRecordID.">  checkMultiBuy_Extract :: AMBIGUOUS EXTRACTION WARNING / INSPECT IN EXTRACTION LOCATION <$pos>"; &logEvent(\*FH_log, $eventType
        , $eventDesc, $guser, $LGDEBUG_FTN);
1173
1174
1175    my $FIND_ADD_CHANGE = 0;
1176
1177    foreach(@GAddressChangeIndicators) { s/[\n\r]//g; next if($_ !~ /\s+/); if($thisPreextractionline =~ /$_/) { $FIND_ADD_CHANGE = 1; } }
1178
1179
1180    if ($FIND_ADD_CHANGE) {
1181
1182            my $t = '.chkAC';
1183            $eventType = 'WRN_FUP'; ($eventRequiresFUP !~ /$t/) ? $eventRequiresFUP .= $t : $eventRequiresFUP ; $FUPcount++; $excpCount++; $eventDesc
        = "\<".$thisRecordID.">  checkMultiBuy_Extract :: POSSIBLE ADD/CHANGE REQUEST DETECTED IN EXTRACTION LOCATION <$pos>"; &logEvent(\*FH_log, $eventType,
        $eventDesc, $guser, $LGDEBUG_FTN);
1184
1185    }
1186
1187    my $FIND_QUESTION = 0;
1188
1189    if($thisPreextractionline =~ /\?\s/) { $FIND_QUESTION = 1; }
1190
1191    if ($FIND_QUESTION) {
1192
1193            my $t = '.chkQst';
1194            $eventType = 'WRN_FUP'; ($eventRequiresFUP !~ /$t/) ? $eventRequiresFUP .= $t : $eventRequiresFUP ; $FUPcount++; $excpCount++; $eventDesc
        = "\<".$thisRecordID.">  checkMultiBuy_Extract :: POSSIBLE ADD/CHANGE OR CARE REQUEST DETECTED IN EXTRACTION LOCATION <$pos>"; &logEvent(\*FH_log, $ev
        entType, $eventDesc, $guser, $LGDEBUG_FTN);
1195
1196    }
1197
1198    my $FIND_AI_ONLY = 0;
1199
1200    if ($exit && scalar(@singlestack) == 2 && scalar(@testsinglestack) == 2) {
1201
1202    ## $x = scalar(@singlestack);
1203    ## $y = scalar(@testsinglestack);
1204
1205    ##print "-- debug -- AI TEST __ singleStackLen = $x __ testsingleStackLen = $y \n";
1206
1207            if(grep /(A)|(I)/, @singlestack) { $FIND_AI_ONLY = 1;}
1208
1209    }
1210
1211    if ($FIND_AI_ONLY) {
1212
1213            my $t = '.vfyTok';
1214            $eventType = 'WRN_FUP'; ($eventRequiresFUP !~ /$t/) ? $eventRequiresFUP .= $t : $eventRequiresFUP ; $FUPcount++; $excpCount++; $eventDesc
        = " \<".$thisRecordID.">  checkMultiBuy_Extract :: AMBIGUOUS EXTRACTION WARNING: \"A\" or \"I\" ONLY DETECTED"; &logEvent(\*FH_log, $eventType, $eventD
        esc, $guser, $LGDEBUG_FTN);
1215
1216    }
1217
1218    my $LIKELY_PARSE_ERRORS = 0;
1219
1220    if ($exit && $thisPreextractionline =~ /[\[\]]+/) { $LIKELY_PARSE_ERRORS = 1; }
1221
1222    if ($LIKELY_PARSE_ERRORS) {
1223
1224            my $t = '.vfyTok';
1225            $eventType = 'WRN_FUP'; ($eventRequiresFUP !~ /$t/) ? $eventRequiresFUP .= $t : $eventRequiresFUP ; $FUPcount++; $excpCount++; $eventDesc
        = " \<".$thisRecordID.">  checkMultiBuy_Extract :: AMBIGUOUS EXTRACTION WARNING: BRACKETS FOUND IN EXTRACTION LOCATION <$pos>"; &logEvent(\*FH_log, $ev
        entType, $eventDesc, $guser, $LGDEBUG_FTN);
```

```
1226      }
1227
1228          ##print "\n-$thisrecordID- debug singlestack -- \n";
1229          ##print @singlestack;
1230          ##print "\n-- debug testsinglestack -- \n";
1231          ##print @testsinglestack;
1232          ##print "\n-$thisrecordID-------------------\n";
1233
1234
1235
1236  return $exit;
1237  }
1238
1239  #-----------------------------------------------
1240  #
1241  #
1242
1243  sub detectBuyTokens {
1244
1245  my($MULTI_BUY_ON,$Gsubj_buyToken,$thissubject,$thismailingID,$thisBodyTop,$thisBodyBottom,$ptrHASHLAYOUT,$ptrSCALARtoplines,$ptrSCALARbottomLines,$GDEBUG)
1246      = @_;
1247
1248  my %LAYOUT            = %$ptrHASHLAYOUT;
1249  my @toplines         = @$ptrSCALARtoplines;
1250  my @bottomLines      = @$ptrSCALARbottomLines;
1251  my $buyTokenPosition = 'none';
1252  $thisBuyStatus       = undef;
1253  @thisItemsselected   = undef;
1254  @thisItemsselectedstack = undef;
1255
1256  my $exit = 0;
1257
1258  ## print "--debug-- buyToken= |$Gsubj_buyToken|\n";
1259
1260  FIND_BUY_TOKENS: {
1261  ##
1262  ## &checkmultiBuy_Extract returns success on a good match/extraction;
1263  ## @thisItemsselected is created/populated at this time as well
1264  ##
1265
1266  ## check $thissubject
1267
1268  ## print "\n\n===========debug-- SUBJECT = |$thissubject \n";
1269
1270          if(!$MULTI_BUY_ON && $thissubject =~ /$Gsubj_buyToken/) {
1271
1272          $buyTokenPosition = ('<'.$buyTokenPosition.'> |'.$thissubject.'|');
1273
1274  ## print "--debug-- thisBuyStatus= |$thisBuyStatus| \n";
1275
1276          last FIND_BUY_TOKENS;
1277          }
1278
1279          if($MULTI_BUY_ON) {
1280
1281                  if(&checkmultiBuy_Extract($thissubject,$thismailingID,' s ',\%LAYOUT,$GDEBUG)) {
1282
1283                          $buyTokenPosition = ' s '; $thisBuyStatus = '<'.$buyTokenPosition.'>';
1284
1285                          last FIND_BUY_TOKENS;
1286                  }
1287
1288
1289
1290  ## check @toplines
1291  ## print "\n\n===========debug-- thisBodyTop = |$thisBodyTop| \n";
1292
1293
```

```
F:\Development\PRJ\PRJ_hbsp_extractOrder\WORKING\proc.pl
Printed at 19:48 on 06 Feb 1999

1294        if(!$MULTI_BUY_ON) {
1295
1296            foreach(@topLines) {
1297
1298                $thisTopLine = $_;
1299
1300                if($thisTopLine =~ /$Gsubj_buyToken/) {
1301
1302                    $buyTokenPosition = ' T ';
1303                    $thisBuyStatus = '<'.$buyTokenPosition.'>'.$thisTopLine;
1304
1305 ## print "--debug-- thisBuyStatus= |$thisBuyStatus| \n";
1306
1307                }
1308
1309            }
1310
1311            last FIND_BUY_TOKENS;
1312        }
1313
1314        if($MULTI_BUY_ON) {
1315
1316            if(&checkMultiBuy_Extract($thisBodyTop,$thisMailingID,' T ',\%LAYOUT,$GDEBUG)) {
1317
1318                $exit =1;
1319
1320                $buyTokenPosition = ' T ';
1321                $thisBuyStatus = '<'.$buyTokenPosition.'>';
1322
1323            }
1324
1325            last FIND_BUY_TOKENS;
1326        }
1327
1328    }
1329
1330 ## check @bottomLines
1331
1332 ## print "\n\n========debug-- thisBottom = |$thisBodyBottom| \n";
1333
1334        if(!$MULTI_BUY_ON) {
1335
1336            foreach(@bottomLines) {
1337
1338                $thisBottomLine = $_;
1339
1340                if($thisBottomLine =~ /$Gsubj_buyToken/) {
1341
1342                    $buyTokenPosition = ' B ';
1343                    $thisBuyStatus = '<'.$buyTokenPosition.'>'.$thisBottomLine;
1344
1345 ## print "--debug-- thisBuyStatus= |$thisBuyStatus| \n";
1346
1347                }
1348
1349            }
1350
1351            last FIND_BUY_TOKENS;
1352        }
1353
1354        if($MULTI_BUY_ON) {
1355
1356            if(&checkMultiBuy_Extract($thisBodyBottom,$thisMailingID,' B ',\%LAYOUT,$GDEBUG)) {
1357
1358                $exit =1;
1359
1360                $buyTokenPosition = ' B ';
1361                $thisBuyStatus = '<'.$buyTokenPosition.'>';
1362
```

```
F:\_Development\PRJ_NPRJ_hbsp_extractOrder--WORKING\procOrder.pl
Printed at 19:48 on 06 Feb 1999

1363                last FIND_BUY_TOKENS;
1364            }
1365        }
1366
1367    }
1368
1369    ## print "--debug-- ==== $i ===== buyTokenPosition= |$buyTokenPosition|\n";
1370
1371    return $buyTokenPosition;
1372    }
1373
1374    #
1375    #----------------------------------------------------------------
1376    #
1377    #
1378    sub extractDedupItemsselectedstack {
1379
1380    my(@thisitemsselected) = @_;
1381
1382    my $thisitemsselectedstack = undef;
1383    my $thisitemsselectedstackseparator = ',';
1384    my $i = undef;
1385
1386    for($i = 0; $i <= $#thisitemsselected; $i++) {
1387
1388        my $item = $thisitemsselected[$i]; # all items must be normalized!
1389
1390        next if($item !~ /\S+?/);
1391
1392        $item=&normalizeSpacing($item);
1393
1394        $thisitemsselectedstack .= ($item.$thisitemsselectedstackseparator);
1395
1396    }
1397
1398    $thisitemsselectedstack =~ s/\,\s*$//;
1399
1400    return $thisitemsselectedstack;
1401    }
1402
1403    #
1404    #----------------------------------------------------------------
1405    #
1406    #
1407    sub getDateStamp {
1408
1409    # get standardized date stamp (UNIX, WINNT)
1410    # returns a date stamp
1411
1412    my($Gplatform,$LGDEBUG_FTN) = @_;
1413
1414    if(!defined($Gplatform)) { $Gplatform = 'WIN'; }
1415    my $GTIME = '';
1416    my $day   = undef;
1417    my $val   = undef;
1418
1419    if ($Gplatform ne 'WIN') { $val = `date '+%Y%m%d%H%M%S-%a'`; }
1420        else {
1421            $val = `date /t`;
1422            chop($val);
1423            $val =~ /(\w+)\s+(\d+)\/(\d+)\/(\d+)\/(\d{2})(\d{2})/;
1424            $val = $5.$2.$3; $day = $1;
1425        }
1426    }
1427
1428    if ($Gplatform eq 'WIN') {
1429        $GTIME = `echo '' | time`;
1430        $GTIME =~ /\:\s*(\d{1,2})\:(\d{2})\:(\d{2})\.(\d{2})/;
1431    }
```

61

```
F:\_Development\PRJ\PRJ_hbsp_extractOrder--WORKIN\procOrder.pl
Printed at 19:48 on 06 Feb 1999

1432         my $leadingDigit = $1;
1433         if (length($leadingDigit) == 1) { $leadingDigit = ('0'.$leadingDigit); }
1434         $val .= $leadingDigit.$2.$3.$4.'.'.$day;
1435
1436     }
1437     if($LGDEBUG_FTN) { print "DEBUG__gdate=".$val."\n"; }
1438
1439     return $val;
1440 }
1441 #
1442 #
1443 # -----------------------------------------------
1444 #
1445 #
1446 sub getfieldkeys {
1447
1448     my($ptrLayout) = @_;
1449     my(%LAYOUT) = %$ptrLayout;
1450
1451     ## prepare and count required fields from body data layout for check against body
1452
1453     my(@fieldkeys)  = undef;
1454     my($k)          = undef;
1455
1456     ##$f = scalar(@fieldkeys); print "--debug-- scalarStartFtn=$f \n";
1457
1458     foreach $k(sort keys %LAYOUT) {
1459
1460         next if($k !~ /^[a-z]{1,2}_/);  ## these elements are not fieldnames but are other CF parameters (name \t value)
1461
1462         ## print "--debug-- k= |$k|\n";
1463
1464         if($k =~ /\S+?/) { push(@fieldkeys,$k); }
1465     }
1466
1467     ##$f = scalar(@fieldkeys); print "--debug-- scalarEndFtn=$f \n";
1468     shift @fieldkeys;
1469
1470     return @fieldkeys;
1471
1472 }
1473 #
1474 # -----------------------------------------------
1475
1476 sub getkeyPos_outfile {
1477
1478     my($ptrHASHrecordTemplate) = @_;
1479
1480     my(%HASHrecordTemplate) = %$ptrHASHrecordTemplate;
1481
1482     my @record                   = undef;
1483     $salutationFieldPos     = -1;
1484     $IDFieldPos             = -1;
1485     $LIDFieldPos            = -1;
1486     $MailingIDFieldPos      = -1;
1487     $firstNameFieldPos      = -1;
1488     $lastNameFieldPos       = -1;
1489     $IxFieldPos             = -1;
1490     $titleFieldPos          = -1;
1491     $stateFieldPos          = -1;
1492     $countryFieldPos        = -1;
1493     $postalCodeFieldPos     = -1;
1494     $emailFieldPos          = -1;
1495     $companyFieldPos        = -1;
1496     $addressFieldPos        = -1;
1497     $departmentFieldPos     = -1;
1498     $cityFieldPos           = -1;
1499     $fromAddressFieldPos    = -1;
1500
```

62

F:\Development\PRJ\PRJ_hbsp_extractOrder-WORKING\hbscOrder.pl
Printed at 19:48 on 06 Feb 1999

```perl
1501  my $i                = 0;
1502
1503  @record = sort(keys(%HASHrecordTemplate));
1504
1505  ##print "--DEBUG---"; print @record; sleep 4;
1506
1507  for($i=0; $i <= $#record; $i++) {
1508
1509      $record[$i] = uc($record[$i]);
1510
1511      CASE: {
1512          if ($record[$i] =~ /FROM\_ADDRESS/)                                    $fromAddressFieldPos = $i; last CASE;
1513          if ($record[$i] =~ /(EMAIL)|(E-MAIL)|(INTERNET_ADDR)|(INTERNETAD)/)    $emailFieldPos       = $i; last CASE;
1514          if ($record[$i] =~ /(COUNTRY)|(CTRY)/)                                 $countryFieldPos     = $i; last CASE;
1515          if ($record[$i] =~ /(ZIP)|(POSTAL)/)                                   $postalCodeFieldPos  = $i; last CASE;
1517          if ($record[$i] =~ /(INST)|(COMPANY)|(ORGANIZATION)/)                  $companyFieldPos     = $i; last CASE;
1518          if ($record[$i] =~ /DEPARTMENT/)                                       $departmentFieldPos  = $i; last CASE;
1519          if ($record[$i] =~ /(ADDR)|(ADDRESS)/)                                 $addressFieldPos     = $i; last CASE;
1520          if ($record[$i] =~ /CITY/)                                             $cityFieldPos        = $i; last CASE;
1521          if ($record[$i] =~ /(CUSTNO)/)                                         $IDFieldPos          = $i; last CASE;
1522          if ($record[$i] =~ /(LISTID)/)                                         $LIDFieldPos         = $i; last CASE;
1523          if ($record[$i] =~ /(PRIORITY)|(MAILING)/)                             $MailingIDFieldPos   = $i; last CASE;
1524          if ($record[$i] =~ /^F\w+NAME.?|(FIRST)/)                              $firstNameFieldPos   = $i; last CASE;
1525          if ($record[$i] =~ /^L\w+NAME.?|(LAST)/)                               $lastNameFieldPos    = $i; last CASE;
1526          if ($record[$i] =~ /^.*X.*$/)                                         $IXFieldPos          = $i; last CASE;
1527          if ($record[$i] =~ /TITLE/)                                            $titleFieldPos       = $i; last CASE;
1528          if ($record[$i] =~ /STATE/ || $record[$i] =~ /^\_CS/)                  $stateFieldPos       = $i; last CASE;
1529      }
1530  }
1531
1532  if (($firstNameFieldPos  == -1)  print STDOUT "[HEADER] [WARNING] \/\w+F|\w+NAME\/ was not found!\n"; }
1533  if (($lastNameFieldPos   == -1)  print STDOUT "[HEADER] [WARNING] \/\w+L|\w+NAME\/ was not found!\n"; }
1534  if (($firstNameFieldPos  == -1)  print STDOUT "[HEADER] [WARNING] \/\w"?X|\w"NAME\/ was not found!\n"; }
1535  if (($titleFieldPos      == -1)  print STDOUT "[HEADER] [WARNING] \/TITLE\/ was not found!\n"; }
1536  if (($stateFieldPos      == -1)  print STDOUT "[HEADER] [WARNING] \/STATE\/ was not found!\n"; }
1537  if (($postalCodeFieldPos == -1)  print STDOUT "[HEADER] [WARNING] \/ZIP|POSTAL\/ was not found!\n"; }
1538  if (($countryFieldPos    == -1)  print STDOUT "[HEADER] [WARNING] \/(COUNTRY)|(CTRY)\/ was not found!\n"; }
1539  if (($emailFieldPos      == -1)  print STDOUT "[HEADER] [WARNING] \/(EMAIL)|(E-MAIL)|(INTERNET_ADDR)|(INTERNETAD)\/ was not found!\n"; }
1540  }
1541
1542
1543
1544  #---------------------------------------------------
1545  #
1546  #
1547
1548  sub getNextDBkey_return {
1549
1550  my($keyFilename, $LGDEBUG_FTN) = @_;
1551
1552      $LOCK_SH = 1;
1553      $LOCK_EX = 2;
1554      $LOCK_NB = 4;
1555      $LOCK_UN = 8;
1556
1557      $POS_BOF = 0;
1558      $POS_CUR = 1;
1559      $POS_EOF = 2;
1560
1561      $/ = undef;
1562      $startkeys = 0;
1563
1564      if(! -s $keyfilename) { `echo $startkeys >> $keyfilename`; }
1565
1566      OPEN_DB: if (!open(KEYS, "+< $keyfilename")) {$exit = 0; $eventType = 'FAIL'; $eventDesc = "getNextDBkey_return::Cannot open: \($keyfilename \)"; &logEven
1567      t(\*FH_log, $eventType, $eventDesc, $eventNumber, $Guser, $LGDEBUG_FTN);}
```

```
F:\Development\PRJ\PRJ_hbsp_extractOrder~WORKING\procOrder.pl
Printed at 19:48 on 06 Feb 1999

1568  flock(KEYS, $LOCK_EX);
1569
1570          my($count)       = <KEYS>;
1571          $nextkey         = $count+1;
1572          seek(KEYS,0,$POS_BOF);
1573          print KEYS $nextkey;
1574
1575  flock(KEYS, $LOCK_UN);
1576  close(KEYS);
1577
1578  return $nextkey;
1579  }
1580  #
1581  #-------------------------------------------
1582  #
1583  #
1584  sub HASHloadFieldMap {
1585
1586  my($sfilename,$LGDEBUG_FTN) = @_;
1587
1589  if (!open(FH, "$filename")) { $eventType = 'FAIL'; $eventDesc = "HASHloadFieldMap::Cannot open: \[$keyfilename \]"; &logEvent(\*FH_log, $eventType, $event
Desc, $Guser, $LGDEBUG_FTN);}
1590
1591  $/ = "\n";
1592
1593  my(@fieldMapTemp) = <FH>;
1594  my($key) = undef;
1595  my($value) = undef;
1596  my(%fieldMap) = undef;
1597
1598  close(FH);
1599
1600          foreach (@fieldMapTemp) {
1601
1602                  my $item = $_;
1603
1604                  next if(($item =~ /\/\//) || ($item !~ /\S+?/)); # comments
1605
1606                  ($key, $value) = split(/[\t\,]/,$item);
1607
1608                  $value =~ s/[\n\r]//g;
1609
1610                  $fieldMap{$key} = $value;
1611
1612          }
1613
1614  return %fieldMap;
1615  }
1616  #
1617  #-------------------------------------------
1618  #
1619  #
1620  sub loadFile_SCALAR {
1621
1622  my($sfilename) = @_;
1623
1625  if (!open(FILE, "$filename")) {$eventType = 'FAIL'; $eventDesc = "loadArray::Cannot open filename: \[$filename\]"; &logEvent(\*FH_log, $eventType, $eventD
esc, $Guser, $LGDEBUG_FTN);}
1626
1627  $/ = "\n";
1628
1629  my(@all) = undef;
1630
1631  while(<FILE>) {
1632
1633          my $buf = $_;
1634          next if($buf !~ /\S+?/ || $buf =~ /\/\//);
```

```
     F:\_Development\PRJPRJ_hbsp_extract\Order-WORKING\procOrder.pl
     Printed at 10:18 on 06 Feb 1999

1635                   push(@all, $buf);
1636 
1637         }
1638 
1639         return @all;
1640 }
1641 #
1642 #
1643 #
1644 #
1645 #
1646 sub logEvent {
1647 
1648 my($FH,$eventType,$eventDesc,$guser,$LGDEBUG_FTN) = @_;
1649 
1650 my $dateNow = &getDatestamp('WIN',$LGDEBUG_FTN);
1651 # print event to STDOUT
1652 
1653 if($eventType eq 'ABORT') {
1654 
1655         $foo = `mapisend -u administrator -p password -r aestes\@e-dialog.com -s "** $dateNow procOrder - Critical Error"  -m "$guser $gclientName $eventT
ype $eventDesc`;
1656         die "Error! Cannot log! ".$eventType.".\t".$eventDesc."\n" ;
1657 }
1658 
1659 if($eventType !~ /x_/) { print $eventType."\t".$eventDesc."\n"; }
1660 
1661         else { $eventType =~ s/_x_//; }
1662 
1663 # write event to activity log
1664 
1665 &writeRecord(\*FH_log,$gfieldDelimiter,$thisLogkey,&getNextDBkey_return($gactivityLogkeysFileName,$LGDEBUG_FTN),\%gactivityLogRecord,'',$LGDEBUG_F
TN);
1666 
1667         if($eventType eq 'FAIL') {
1668         $foo = `mapisend -u administrator -p password -r aestes\@e-dialog.com -s "** $dateNow procOrder - Critical Error"  -m "$guser $gclientName
$eventType $eventDesc`;
1669                       exit;
1670         }
1671 }
1672 #
1673 #
1674 #
1675 #
1676 #
1677 sub normalizeSpacing {
1678 
1679 my($dirty) = @_;
1680 
1681 $dirty =~ s/\s{2,}/$space/g; $dirty =~ s/\s+$//; $dirty =~ s/^\s+//;
1682 $clean = $dirty;
1683 
1684 return $clean
1685 }
1686 #
1687 #
1688 #
1689 #
1690 #
1691 sub queryUPDATE_DB {
1692 
1693 my($ptrLAYOUT,$thisqueryKey, $thisqueryFieldName, $thisqueryFieldName, $fieldName, $fieldvalue, $fieldName1, $fieldvalue1, $RECORD_RECOVERED, $LGDEBUG_FTN) = @_;
1694 
1695 #print "** DSN = $LAYOUT('thisMailingDSN') \n";
1696 
1697 my(%LAYOUT) = %$ptrLAYOUT;
1698 my $RS = undef;
1699 my $Conn = undef;
```

65

```
F:\Development\PRJNPRJ-htsp\extractorder-WORK\extractorder.pl
Printed at 19:48 on 06 Feb 1999

1701 my $errors = undef;
1702 my @fieldNames = undef;
1703 my $PID = -1;
1704 my $recCount = 0;
1705 my $EXIT = 1;
1706
1707 # construct/open DSN
1708
1709 my $conn = undef;
1710
1711 $conn = new Win32::OLE("ADODB.Connection");
1712 $conn->open($LAYOUT{'thisMailingDSN'}, "", "");
1713
1714 my(%DB) = undef;
1715
1716 # load data into recordset
1717
1718     my $mailingTable = $LAYOUT{'thisMailingTable'};
1719     my %foo = undef;
1720
1721 # look to see if update record exists, $thisquerykey
1722
1723 unless($RECORD_RECOVERED) {
1724
1725     my $RECORD_EXISTS = 1;
1726     %foo = &querySELECT_DB(%LAYOUT,$thisquerykey,$LGDEBUG,$LGDEBUG_FTN);
1727     if($foo{'_exit_'} < 1) { $RECORD_EXISTS = 0; $EXIT = 0; return $EXIT; }
1728
1729 }
1730     $thisquery = "UPDATE $mailingTable SET $fieldName = \'$fieldvalue\', $fieldName1 = \'$fieldvalue1\'  WHERE $thisqueryfieldname = \'$thisquerykey\'"
1731 ;
1732     $RS = $conn->Execute($thisquery);
1733
1734 ##%foo = %$RS;
1735 ##foreach(keys(%foo)) { print "--key = $_ -- value = $foo{$_} \n"; }
1736
1737     if(!$RS) {
1738
1739         $errors = $conn->Errors();
1740         print "errors:\n";
1741         foreach $error (keys %$errors) {
1742             print $error->{description}, "\n";
1743         }
1744         $eventType = 'WRN_FUP'; $eventRequiresFUP .= (($eventRequiresFUP =~ /badMDB/)?'':'.badMDB'); $eventDesc = " \<".$thisRecordID."\>"        ($LAYO
    urf('thisMailingDSN')) queryUPDATE_DB :: Cannot create RS: \[$thisquery\]"; &logevent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
1745         $EXIT = 0;
1746         return $EXIT;
1747     }
1748
1749
1750     $eventType = 'INFO'; $eventDesc = " \<".$thisRecordID."\>      ($LAYOUT('thisMailingDSN')) queryUPDATE_DB :: UPDATE SUCCESSFUL: \[$thisquery\]"; &log
    Event(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
1751
1752 return $EXIT;
1753 }
1754 #--------------------------------------------------
1755 #
1756 #
1757
1758 sub querySELECT_DB {
1759
1760 my($ptrLAYOUT,$thisqueryfieldname,$thisquerykey,$LGDEBUG,$LGDEBUG_FTN) = @_;
1761
1762
1763 ##print "** DSN = $LAYOUT{'thisMailingDSN'} \n";
1764
1765 my(%LAYOUT) = %$ptrLAYOUT;
1766 my $RS = undef;
```

66

```
F:\_Development\PRJ\PRJ_1\hbsp_extractOrder_WORKING\procOrder.pl
Printed at 19:48 on 06 Feb 1999

1767  my $conn = undef;
1768  my $errors = undef;
1769  my @fieldNames = undef;
1770  my $PID = -1;
1771  my $reccount = 0;
1772
1773  # construct/open DSN
1774
1775  my $conn = undef;
1776
1777  $conn = new win32::OLE("ADODB.Connection");
1778  $conn->open($LAYOUT('thisMailingDSN'), "", "");
1779
1780  my(%DB) = undef;
1781  $DB{'_exit_'} = 1;
1782
1783  # Load data into recordset
1784
1785  my $mailingTable = $LAYOUT('thisMailingTable');
1786
1787  $thisquery = "SELECT * FROM $mailingTable WHERE $thisqueryFieldname = \"$thisquerykey\"";
1788
1789  $RS = $conn->Execute($thisquery);
1790
1791  if(!$RS) {
1792
1793      $errors = $conn->Errors();
1794      print "Errors:\n";
1795      foreach $error (keys %errors) {
1796          print $error->{Description}, "\n";
1797      }
1798      $eventType = 'FAIL'; $eventDesc = " \<".$thisRecordID."\>    ($LAYOUT('thisMailingDSN')) querySELECT_DB :: Cannot create RS: \[$thisquery]"; &loge
vent(\*FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);
1799  }
1800
1801  my $i = undef;
1802
1803  for($i=0; $i < $RS->Fields->Count; $i++) { $fieldNames[$i] = $RS->Fields->Item($i)->Name; }
1804
1805  my $RESULT_FLAG = 1;
1806
1807  if($RS->EOF) { $RESULT_FLAG = 0; }
1808  $RS -> MoveNext;
1809
1810  if(!$RS->EOF) { $RESULT_FLAG = -1; }
1811  $RS -> MovePrevious;
1812
1813  if($RESULT_FLAG < 1) {
1814
1815      (($eventRequiresFUP =~ /badMDB/)?'':'badMDB'); $FUPcount++; $eventType = 'ERR_FUP'; $eventDesc = " \<".$thisRecordID."\>    ($LAYOUT('thisMailingD
SN')) querySELECT_DB :: Bad Lookup ($RESULT_FLAG): \[$thisquery]"; &logevent(\*FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);
1816
1817      $DB{'_exit_'} = $RESULT_FLAG;
1818
1819  return %DB;
1820  }
1821
1822  while ( !$RS->EOF ) {
1823
1824      for($i=0; $i < $RS ->Fields->Count; $i++) {
1825
1826          $DB{$fieldNames[$i]} = uc($RS->Fields->Item($i)->value);
1827
1828      }
1829      $RS->MoveNext; $recCount++;
1830
1831  }
1832  $conn->Close;
1833
```

```
      F:\_Development\PRJRB4_hbsp_extractOrder--WORKIN\procOrder.pl
      Printed at 16:48 on 06 Feb 1999
1834  return %DB;
1835  }
1836  #
1837  #
1838  # ------------------------------------------------
1839  #
1840
1841  sub read_DSNtoAOH {
1842
1843  my($DSN,$LGDEBUG_FTN) = @_;
1844
1845  ##print "** DSN = $DSN \n";
1846
1847  $RS = undef;
1848  $conn = undef;
1849  $errors = undef;
1850  @fieldNames = undef;
1851  $PID = -1;
1852  $recCount = 0;
1853
1854  # construct/open DSN
1855
1856  $conn = new win32::OLE("ADODB.Connection");
1857  $conn->Open($DSN,"","");
1858  my(@FILE) = undef;
1859
1860  # load data into recordset
1861
1862  $thisquery = "SELECT * FROM Email";
1863
1864  $RS = $conn->Execute($thisquery);
1865
1866  if(!$RS) {
1867      $errors = $conn->Errors();
1868      print "errors:\n";
1869      foreach $error (keys %$errors) {
1870          print $error->{Description}, "\n";
1871      }
1872      $eventType = 'FAIL'; $eventDesc = "read_DSNtoAOH::Cannot create RS: \[$thisquery\]"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_
      FTN);
1873  }
1874
1875  my $i = undef;
1876  for($i=0; $i < $RS->Fields->Count; $i++) { $fieldNames[$i] = $RS->Fields->Item($i)->Name; }
1877
1878
1879  while ( !$RS->EOF ) {
1880
1881      for($i=0; $i < $RS ->Fields->Count; $i++) {
1882
1883          $FILE[$recCount]{$fieldNames[$i]}= $RS->Fields->Item($i)->value;
1884
1885      }
1886
1887      $RS->MoveNext; $recCount++; if ($recCount % 20 == 0) { print STDOUT "+"; }
1888
1889
1890  $conn->Close;
1891  }
1892  return @FILE;
1893  }
1894  #
1895  # ------------------------------------------------
1896  #
1897  #
1898
1899  sub removeTabs {
1900
1901  my($tabs) = @_;
```

```
P:\_Development\PRJ\PRJ_hbsp_extractOrder--WORKING\procOrder.pl
Printed at 18:48 on 06 Feb 1999

1902  $tabs = s/\t//g;
1903  return $tabless
1904
1905
1906  }
1907
1908  # ---------------------------------------------
1909  #
1910  #
1911
1912
1913  sub subjectSynonymnLookup {
1914
1915    my($subject,$ptrHASHLAYOUT) = @_;
1916    my(%LAYOUT) = %$ptrHASHLAYOUT;
1917    my($thisMailingID) = -1;
1918
1919      $subject =~ /\s*([\w\+\-]{1,3})\s*/;
1920      $thisSubjectMailingIDToken = $1;
1921
1922      if(defined($LAYOUT{$thisSubjectMailingIDToken})) { $thisMailingID = $LAYOUT{$thisSubjectMailingIDToken}; }
1923
1924      ## print "-- debug -- ATTEMPT MAILING ID RECOVERY... TOKEN= |$thisSubjectMailingIDToken|... RESULT MAILINGID= |$thisMailingID|\n";
1925
1926    return $thisMailingID
1927  }
1928
1929  # ---------------------------------------------
1930  #
1931  #
1932
1933  sub writeRecord {
1934
1935  # takes FH by ref; has _e_ feature for values
1936
1937    my($FH,$GfieldDelimiter,$logkey,$ptrHASHrecord,$RECORD_TYPE, $LGDEBUG_FTN) = @_;
1938
1939    flock($FH, $LOCK_EX);
1940
1941    my $fileEmpty = 0;
1942    if(-z $FH) {$fileEmpty = 1;}
1943
1944    %HASHrecord = %$ptrHASHrecord;
1945
1946    $space = " ";
1947    $exit = 1;
1948
1949    $LOCK_SH = 1;
1950    $LOCK_EX = 2;
1951    $LOCK_NB = 4;
1952    $LOCK_UN = 8;
1953
1954    my $record = undef;
1955    my $header = undef;
1956    my $key = undef;
1957
1958    foreach $key(sort(keys(%HASHrecord))) {
1959
1960      $value = $HASHrecord{$key};
1961
1962      if($fileEmpty) { $key =~ s/[a-z]{1,2}_//; $header .= (uc($key).$Gfielddelimiter); }
1963
1964      if($value =~ /^_e_(.{2,})$/) { $value = eval($1); }
1965
1966      $record .= ($value.$Gfielddelimiter);
1967
1968    }
1969    chop($record);                                    # remove last field delimiter
1970
```

69

Page 31

```
1971  if($fileEmpty) { chop($header); print $FH $header. "\n"; }
1972  if($ADD_CHANGE_ON && $RECORD_TYPE ne 'BODY') { $record = &canonicalize($record); }
1973  print $FH $record. "\n";
1974
1975  flock($FH, $LOCK_UN);
1976  }
1977  __END__
1978
1979
1980
```

70

Claims

1   1.   A machine-based method comprising

2   analyzing an e-mail message to derive response

3 information concerning a commercial transaction, and

4 based on the derived information, and

5   automatically generating commercial transaction data

6 in a format that is usable to automatically complete the

7 commercial transaction.

1   2.   The method of claim 1 in which the commercial

2 transaction comprises an order for a product or service.

1   3.   The method of claim 1 in which the e-mail

2 message comprises at least part of an e-mail sent to a

3 customer and responses of the customer to the e-mail.

1   4.   The method of claim 1 in which the automatic

2 completion of the commercial transaction comprises order

3 fulfillment.

1   5.   A machine-based method comprising

2   sending an e-mail message to a customer offering a

3 product or service for sale, the e-mail message comprising

4 locations for response by the customer indicating his

5 intention to order the product or service,

6   receiving from the customer an e-mail message that

7 includes the response,

8   based on the received e-mail, automatically

9 generating order information in a format usable

10 automatically by an order fulfillment system to cause the

11 order to be filled.

1   6.   A machine-based method comprising

2   analyzing an e-mail message to derive response

3 information concerning a commercial transaction,

4   automatically identifying response information which

5 requires resolution of an issue with the source of the e-

6 mail message, and

7   automatically managing an e-mail dialog with the
8   source to resolve the issue.

1   7.   The method of claim 6 in which at least some of
2   the e-mail dialog is performed automatically.

1   8.   Software guided interactive e-mail dialogs to
2   resolve, on behalf of a vendor, customer issues that occur
3   in direct response e-mails that are automatically identified
4   as requiring a dialog.

1   9.   A machine-based method comprising
2   automatically sorting e-mail messages, based on
3   response information contained in the messages, into e-mail
4   messages that can be processed automatically to generate
5   commercial transactions, e-mail messages in which the
6   response information is inadequate to permit generation of
7   commercial transactions, and e-mail messages that may be
8   subjected to exception handling to yield information that is
9   sufficient to generate commercial transactions.

1   10.   A machine-based method comprising
2   analyzing an e-mail message to derive response
3   information concerning a commercial transaction, and
4   automatically generating a confirmatory e-mail
5   message to the source of the e-mail message confirming that
6   the commercial transaction has been or will be completed.

1   11.   A machine-based method comprising
2   receiving inbound e-mail messages that result from
3   corresponding outbound e-mail messages associated with a
4   marketing program, the inbound messages containing response
5   information, each of the outbound messages being associated
6   with a distinct piece of the marketing program, and
7   automatically associating the response information
8   in each of the inbound messages with the corresponding
9   distinct piece of the marketing program.

72

1   12.   The method of claim 11 in which the piece
2   comprises a marketing campaign or a marketing flight.

1   13.   The method of claim 11 in which the inbound
2   messages contain information that links them to the
3   corresponding outbound messages, and the associating step
4   uses the link information.

1   14.   The method of claim 13 further comprising
2   automatically parsing the inbound messages for order
3   information.

1   15.   A machine-based method comprising
2   sending outbound e-mail messages associated with
3   commercial transactions,
4   storing information related to each of the outbound
5   messages in a database, the information being useful for
6   completing the commercial transactions, the information not
7   being contained in the outbound messages,
8   analyzing inbound e-mail messages that result from
9   the outbound messages and that contain response information
10 useful in completing the commercial transactions, and
11   automatically merging the response information with
12 corresponding information in the database for use in
13 completing the transactions.

1   16.   A machine-based method comprising
2   sending outbound e-mail messages associated with
3   commercial transactions,
4   storing information related to each of the outbound
5   messages in a database, the information being useful for
6   completing the commercial transactions, the information not
7   being contained in the outbound messages,
8   analyzing inbound e-mail messages that result from
9   the outbound messages and that contain response information
10 useful in completing the commercial transactions,

73

11   identifying inbound e-mail messages that cannot be
12   processed automatically to generate the commercial
13   transactions, and
14   using the database information to assist in
15   exception handling of the identified inbound messages.

WILLIAM HERP　　　　　　　　　　　　　　　　　　　　　　　　　　10

FROM:　　　　HARVARD BUSINESS SCHOOL PUBLISHING
SENT:　　　　WEDNESDAY, DECEMBER 16, 1998 9:19 PM
TO:　　　　　WILLIAM HERP
SUBJECT:　　 = NEW INSIGHTS FROM HARVARD BUSINESS REVIEW

HARVARD BUSINESS SCHOOL PUBLISHING CORPORATION
BOSTON, MASSACHUSETTS USA

THURSDAY, DECEMBER 17, 1998

DEAR WILLIAM HERP,

ON THURSDAY, DECEMBER 3RD, WE WROTE YOU REGARDING A SPECIAL OFFER ON THE
HARVARD BUSINESS REVIEW PAPERBACK SERIES. SINCE WE HAVE NOT HEARD BACK, WE
WANTED TO FOLLOW UP BEFORE THIS SPECIAL OFFER CLOSES. IF YOU ARE SIMPLY NOT
INTERESTED, WE APOLOGIZE FOR THE INTRUSION. BELOW PLEASE FIND THE ORIGINAL
OFFER IN ITS ENTIRETY.

THE HARVARD BUSINESS REVIEW PAPERBACK SERIES BRINGS YOU THE LATEST AND MOST
SIGNIFICANT THINKING ON TODAY'S MOST PRESSING MANAGEMENT CHALLENGES. THESE
INSIGHTFUL COLLECTIONS ARE THE DEFINITIVE RESOURCE FOR PROFESSIONALS.

EACH TITLE:
 + PROVIDES A BROAD UNDERSTANDING OF AN ISSUE
 + IS CLEARLY WRITTEN AND, IN MANY CASES, DRAWS UPON REAL COMPANY EXAMPLES
 + HELPS YOU CONSTRUCT A USEFUL CONCEPTUAL FRAMEWORK FOR DECISION-MAKING
AND IMPLEMENTATION
 + CONTAINS EIGHT ARTICLES FROM HARVARD BUSINESS REVIEW

◇ EACH PAPERBACK IS $19.95 PLUS SHIPPING AND HANDLING ◇

〈〈

TO ORDER ONE OR MORE OF THESE PAPERBACKS, SIMPLY REPLY TO THIS MESSAGE AND
NOTE THE LETTER (A-F) OF THE HARVARD BUSINESS REVIEW PAPERBACK YOU WOULD LIKE
TO RECEIVE. PLEASE TYPE THE LETTERS IN THE 1ST LINE OF THE BODY OF YOUR REPLY
E-MAIL. SHIPPING AND HANDLING CHARGES WILL BE APPLIED TO EACH ORDER.

YOUR CHOICES (DETAILED BELOW) ARE:

A: HARVARD BUSINESS REVIEW ON CHANGE PAPERBACK
B: HARVARD BUSINESS REVIEW ON KNOWLEDGE MANAGEMENT PAPERBACK
C: HARVARD BUSINESS REVIEW ON STRATEGIES FOR GROWTH PAPERBACK
D: HARVARD BUSINESS REVIEW ON MEASURING CORPORATE PERFORMANCE PAPERBACK
E: HARVARD BUSINESS REVIEW ON LEADERSHIP PAPERBACK

F: THE EXECUTIVE COLLECTION - ALL FIVE TITLES FOR $89

OR IF YOU PREFER, CALL 1-800-668-6780 (617-496-1449 OUTSIDE THE U.S.)
MON. - FRI. 8 A.M. - 6 P.M. EST. PLEASE BE SURE TO MENTION PRIORITY CODE 3202.

〉〉〉

12

14

FIG. 1A

2/6

A: ** HARVARD BUSINESS REVIEW ON CHANGE **
PROVIDES LANDMARK IDEAS TO HELP YOU UNDERSTAND THE BEST WAYS FOR YOUR
ORGANIZATION TO MANAGE CHANGE. INCLUDES ARTICLES BY JOHN KOTTER AND MORE.
(240 PP/#8842/$19.95)

B. ** HARVARD BUSINESS REVIEW ON KNOWLEDGE MANAGEMENT **
HIGHLIGHTS THE LEADING-EDGE THINKING AND PRACTICAL APPLICATIONS ON HOW COMPANIES
GENERATE, COMMUNICATE, AND LEVERAGE KNOWLEDGE ASSETS. INCLUDES ARTICLES BY PETER
DRUCKER, JOHN SEELY BROWN, AND MORE.
(240 PP/#8818/$19.95)

C: ** HARVARD BUSINESS REVIEW ON STRATEGIES FOR GROWTH **
PRESENTS THE LATEST TACTICS FOR HELPING MANAGERS FIND AND EXPLOIT THE BEST
OPPORTUNITIES FOR GROWTH AND PROFITABILITY. INCLUDES ARTICLES BY ARIE DE GEUS,
JEFFREY RAYPORT, AND MORE.
(240 PP/#8850/$19.95)

D: ** HARVARD BUSINESS REVIEW ON MEASURING CORPORATE PERFORMANCE **
OFFERS INSIGHT ON WHAT YOU NEED TO MEASURE AND HOW PERFORMANCE MEASURES CAN
ALIGN AN ORGANIZATION AND BOOST PRODUCTIVITY. INCLUDES ARTICLES BY PETER DRUCKER,
ROBERT KAPLAN AND DAVID NORTON, AND MORE.
(240 PP/#8826/$19.95)

E: ** HARVARD BUSINESS REVIEW ON LEADERSHIP **
PRESENTS PROVEN FUNDAMENTALS OF LEADERSHIP AND CHALLENGES MANY LONG-HELD
ASSUMPTIONS ABOUT THE TRUE SOURCES OF POWER AND AUTHORITY. INCLUDES ARTICLES BY
JOHN KOTTER, JOSEPH BADARACCO, JR., AND MORE.
(240 PP/#8834/$19.95)

F: PURCHASE THE EXECUTIVE COLLECTION (INCLUDES ALL FIVE TITLES) FOR JUST $89- A SAVINGS
OF MORE THAN 10%. (PRODUCT #7419BN)

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

FIG. 1B

3/6

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

◇EACH PAPERBACK IS $19.95 PLUS SHIPPING AND HANDLING ◇

IMPORTANT NOTE FOR OUR CUSTOMERS OUTSIDE THE U.S.: PURCHASERS ARE
RESPONSIBLE FOR ALL DUTIES, TAXES, BROKERAGE FEES, AND/OR IMPORT FEES IMPOSED
BY THE COUNTRY OF IMPORT. SHIPPING AND HANDLING CHARGES WILL BE APPLIED TO
YOUR ORDER. DELIVERY TO CANADA: $14.00 FOR THE FIRST TITLE, $2.00 FOR EACH
ADDITIONAL TITLE. INTERNATIONAL DELIVERY OUTSIDE NORTH AMERICA: $20.00 FOR THE
FIRST TITLE, $5.00 FOR EACH ADDITIONAL TITLE.

PLEASE ALSO REVIEW AND UPDATE THE ADDRESS INFORMATION BELOW SO THAT WE CAN
PROCESS YOUR REQUEST PROMPTLY.

```
FIRST NAME:        [WILLIAM]
LAST NAME:         [HERP]
TITLE:             [PRESIDENT]
COMPANY:           [E-CARE GROUP INC.]
DEPARTMENT:        []
ADDRESS1:          [1646 MASSACHUSETTS AVE.]
ADDRESS2:          []
ADDRESS3:          []

CITY:              [LEXINGTON]
PROVINCE/STATE:    [MA]
POSTAL/ZIP CODE:   [2173]
COUNTRY:           []
PHONE:             [    ]
FAX:               [    ]
EMAIL:             [WHERP@E-CARE.COM]
```
— 16

[[891270||3202|]]

FIG. 1C

WILLIAM HERP
_____

FROM: HARVARD BUSINESS SCHOOL PUBLISHING
SENT: MONDAY, FEBRUARY 08, 1999 8:25 PM
TO: WILLIAM HERP
SUBJECT: ** A FREE NO-OBLIGATION TRIAL FROM HARVARD BUSINESS REVIEW

FROM THE DESK OF LAURA WINIG
HARVARD BUSINESS SCHOOL PUBLISHING CORPORATION
BOSTON, MASSACHUSETTS

MONDAY, FEBRUARY 8, 1999

=_____=

~INTRODUCING~

BENCHMARKING A NEW THREE-PART VIDEO SERIES FROM HARVARD BUSINESS SCHOOL PUBLISHING
CORPORATION

TO TAKE *BENCHMARKING* FOR A NO-OBLIGATION 14-DAY TEST DRIVE, SIMPLY REPLY TO THIS E-MAIL
WITH THE WORD "YES" IN THE SUBJECT LINE

=_____=

DEAR WILLIAM HERP:

INTERESTED IN UNEARTHING NEW IDEAS AND UNCONVENTIONAL SOLUTIONS FOR THE CHALLENGES
FACING YOUR COMPANY? HERE AT THE PUBLISHING ARM OF HARVARD BUSINESS SCHOOL, WE'VE CREATED
AN EXCITING NEW PROGRAM THAT CAN SHOW YOU HOW SOME LEADING COMPANIES USE BENCHMARKING
-- STUDYING AND EMULATING TOP PERFORMERS INSIDE, AND OUTSIDE, THEIR INDUSTRIES -- TO ELIMINATE
LONG-STANDING PROBLEMS AND BECOME TOP PERFORMERS.

DISCOVER HOW NEW PRACTICES CAN BE APPLIED TO YOUR ORGANIZATION -- WITH IMPRESSIVE AND
MEASURABLE RESULTS -- IN BENCHMARKING, AN INNOVATIVE THREE-PART VIDEO SERIES. WE'LL TAKE
YOU DEEP INSIDE PROFILED COMPANIES SUCH AS MOBIL OIL, GTE, AND SUNHEALTH TO LEARN HOW THEY
IDENTIFIED "BEST OF CLASS" COMPANIES TO BENCHMARK IN ORDER TO IMPROVE THEIR OWN
PERFORMANCE.

YOU'LL SEE HOW BENCHMARKING CAN GIVE YOUR TEAM A COMMON RALLYING POINT AND MOTIVATE
COORDINATED ACTION. YOU'LL LEARN HOW TO IDENTIFY PROCESSES TO BENCHMARK, HOW TO FIND THE
RIGHT PARTNER, AND HOW TO INITIATE THE FIRST STEPS (EVEN ON A LIMITED BUDGET). YOU'LL FIND OUT
HOW TO IDENTIFY NOVEL OPPORTUNITIES, HOW TO STRUCTURE YOUR EFFORTS FOR SUCCESS, EVEN
PROPER BENCHMARKING ETIQUETTE. EACH CONCEPT IS CLEARLY EXPLAINED AND ILLUSTRATED TO
FACILITATE IMPLEMENTATION.

BENCHMARKING FOR CONTINUOUS IMPROVEMENT, BENCHMARKING CORE PROCESSES, AND
BENCHMARKING OUTSIDE THE BOX BRING YOU FIRSTHAND COMMENTARY FROM SENIOR EXECUTIVES,
INDUSTRY EXPERTS, AND FRONT-LINE PERSONNEL IN A FAST-PACED DOCUMENTARY STYLE THAT
GENERATES INTEREST, UNDERSTANDING, AND ENTHUSIASM FOR THESE IMPORTANT IDEAS. THESE VIDEOS
WILL STIMULATE DISCUSSION AND PROVIDE GUIDELINES TO HELP YOU DEVELOP AN ACTION PLAN FOR
YOUR ORGANIZATION.

MAY I SEND YOU BENCHMARKING FOR A FREE, NO-OBLIGATION TRIAL? SIMPLY REPLY TO THIS E-MAIL
WITH THE WORD "YES" IN THE SUBJECT LINE AND WE'LL SEND YOU THE PROGRAM TO TRY WITH OUR
COMPLIMENTS. WE'LL SEND YOU THIS INNOVATIVE SERIES RIGHT AWAY. AFTER 14 DAYS, WE WILL MAIL
YOU AN INVOICE FOR $1190 (A SAVINGS OF $595 VERSUS THE INDIVIDUAL VIDEO PRICE OF $595 EACH).

IF YOU ARE NOT COMPLETELY SATISFIED WITH BENCHMARKING SIMPLY RETURN IT TO US. YOU WILL OWE
NOTHING. WHY WAIT TO LEARN HOW SUCCESSFUL CHANGE MANAGEMENT CAN DRAMATICALLY ENHANCE
YOUR ORGANIZATION'S PERFORMANCE?

SINCERELY,

LAURA WINIG

FIG. 2A

5/6

DIRECTOR

P.S. IF YOU PREFER, PRINT OUT THIS INVITATION, INITIAL IT AT THE TOP, (PLEASE VERIFY YOUR SHIPPING ADDRESS IS CORRECT AS LISTED ABOVE -- WE MUST HAVE A STREET ADDRESS FOR SHIPMENT) AND FAX IT TO 617-496-1029, OR SIMPLY CALL 1-800-668-6780. PLEASE BE SURE TO MENTION PRIORITY CODE 3275.

CONTACT INFORMATION

BELOW PLEASE FIND THE CONTACT INFORMATION WE CURRENTLY HAVE ON FILE. IF THIS INFORMATION IS NOT CORRECT, PLEASE MAKE YOUR EDITS BETWEEN THE APPROPRIATE BRACKETS AND RETURN - VERBATIM - AS PART OF YOUR REPLY E-MAIL. PLEASE INDICATE ANY ADDRESS CHANGE BY INCLUDING THE WORDS 'ADDRESS CHANGE" AT THE TOP OF YOUR ORDER-REPLY.

IF YOU WISH TO UNSUBSCRIBE FROM SPECIAL OFFER MAILINGS, PLEASE REPLY TO THIS E-MAIL MESSAGE WITH THE WORD "UNSUB" AT THE TOP OF YOUR REPLY.

BILLING ADDRESS

BILL FIRST NAME:	[WILLIAM]
BILL LAST NAME:	[HERP]
BILL TITLE:	[PRESIDENT]
BILL COMPANY:	[E-CARE GROUP INC.]
BILL DEPARTMENT:	[ ]
BILL ADDRESS1:	[1646 MASSACHUSETTS AVE.]
BILL ADDRESS2:	[ ]
BILL ADDRESS3:	
BILL CITY:	[LEXINGTON
BILL PROVINCE/STATE:	[MA]
BILL POSTAL/ZIP CODE:	[02173]
BILL COUNTRY:	[ ]
BILL PHONE:	[ ]
BILL FAX:	[ ]
BILL EMAIL:	[WHERP@E-CARE.COM]

SHIPPING ADDRESS (IF DIFFERENT)

SHIP FIRST NAME:	[ ]
SHIP LAST NAME:	
SHIP TITLE:	
SHIP COMPANY:	
SHIP DEPARTMENT:	
SHIP ADDRESS1:	
SHIP ADDRESS2:	
SHIP ADDRESS3:	
SHIP CITY:	
SHIP PROVINCE/STATE:	
SHIP POSTAL/ZIP CODE:	
SHIP COUNTRY:	
SHIP PHONE:	
SHIP FAX:	
SHIP EMAIL:	[
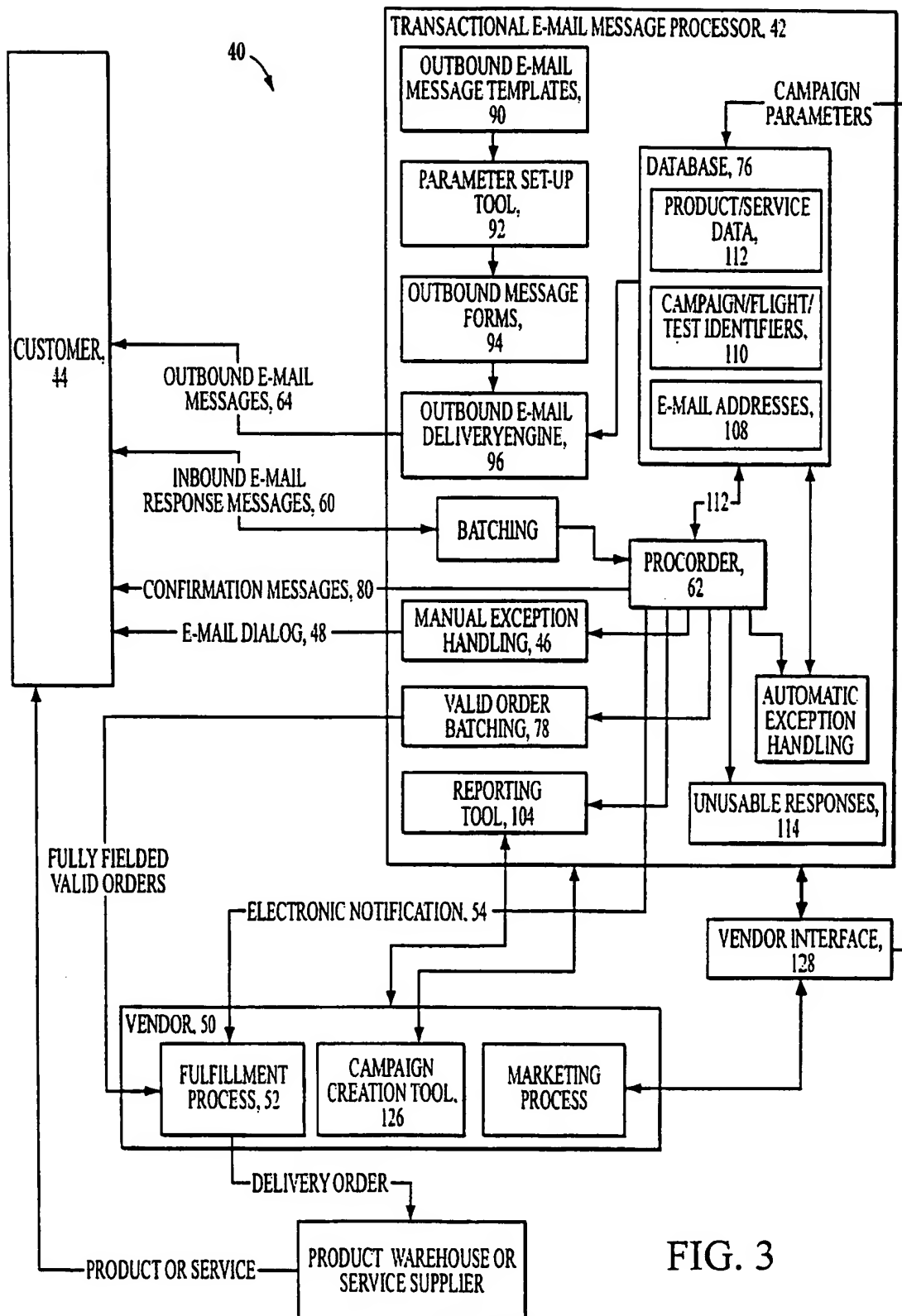
~18

[[891270||3275|]]   }~66

68   70   72

FIG. 2B

6/6

FIG. 3

INTERNATIONAL SEARCH REPORT

A.   CLASSIFICATION OF SUBJECT MATTER

IPC(7)   :G06F 17/60
US CL   : 705/1

According to International Patent Classification (IPC) or to both national classification and IPC

B.   FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. :   705/1,709/206,207,379/193,707/505

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

DIALOG, WEST,EAST

C.   DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5,724,424 A (GIFFORD) 03 Mar 1998, Fig 1[68,64,66,67,62,200]; col 4 lines 48-53; col 5 lines 22-29; Fig.3[5,6,7]; col 5 lines 18-19;col 5 lines 34-37;Fig 5[18]; Fig 4[15]; col 5 lines 49-57; Fig 6[19-25];col 5 lines 29-44;Fig 4[8-17];Fig 2[1-3];Fig 3[3];col 5 line 25-col 6 line 5];Fig. 5;col 6 lines 20-29;col 6 lines 39-42;Fig.3[3]	1-16

☐ Further documents are listed in the continuation of Box C.   ☐   See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	
"E"	earlier document published on or after the international filing date	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"O"	document referring to an oral disclosure, use, exhibition or other means	
"P"	document published prior to the international filing date but later than the priority date claimed	"&"	document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
19 AUGUST 2000	02 OCT 2000

Name and mailing address of the ISA/US	Authorized officer
Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231	Tod Swann
Facsimile No. (703) 305-3230	Telephone No. (703)

Form PCT/ISA/210 (second sheet) (July 1998)★